Infrared Data Association

'IrCOMM': Serial and Parallel Port Emulation over IR (Wire Replacement)

Version 1.0

7 November, 1995

Counterpoint Systems Foundry, Inc.
Hewlett-Packard Company
Lexmark International, Inc.
Sharp Corporation
NTT Corporation
Nokia

Authors:

David W. Suvak (Hewlett-Packard Company and Counterpoint Systems Foundry)
Patrick J. Megowan (Hewlett-Packard Company and Counterpoint Systems Foundry)
Lloyd Young (Lexmark)
Takashi Kondo (Sharp Corporation)
Masahiro Esashi (Sharp Corporation)
Mitsuji Matsumoto (NTT Corporation)
Ryuichi SAITO (NTT Corporation)
Petri Nykanen (Nokia)

Contributors:

Paul McClellan, Stuart Williams (Hewlett-Packard Company)
Frank Novak (IBM)
Wassef Haroun (Microsoft)
Gunnar Sjolund, Stephen Rybicki (Puma Technology)
Osamu Tsumori (Sharp Corporation)

Document Status: Version 1.0

INFRARED DATA ASSOCIATION (IrDA) - NOTICE TO THE TRADE -

SUMMARY:

Following is the notice of conditions and understandings upon which this document is made available to members and non-members of the Infrared Data Association.

- Availability of Publications, Updates and Notices
- Full Copyright Claims Must be Honored
- Controlled Distribution Privileges for IrDA Members Only
- Trademarks of IrDA - Prohibitions and Authorized Use
- No Representation of Third Part Rights
- Limitation of Liability
- Disclaimer of Warranty
- Certification of Products Requires Specific Authorization from IrDA after Product Testing for IrDA Specification Conformance

IrDA PUBLICATIONS and UPDATES:

Single issues of each IrDA publication, including notifications, updates, and revisions, are distributed to IrDA members in good standing during the course of each year as a benefit of annual IrDA membership. Additional copies are available to IrDA members for a fee which covers the cost of reproduction and distribution. Annual subscriptions of IrDA publications are available to non-IrDA members for a pre-paid fee. Requests for publications, membership applications or more information should be addressed to: Infrared Data Association, P.O. Box 3883, Walnut Creek, California, U.S.A. 94598; or e-mail address: jlaroche@netcom.com; or by calling John LaRoche at (510) 943-6546 or faxing requests to (510) 934-5241.

COPYRIGHT:

1. Prohibitions: IrDA claims copyright in all IrDA publications. Any unauthorized reproduction, distribution, display or modification, in whole or in part, is strictly prohibited.

2. Authorized Use: Any authorized use of IrDA publications (in whole or in part) is under NONEXCLUSIVE USE LICENSE ONLY. No rights to sublicense, assign or transfer the license are granted and any attempt to do so is void.

DISTRIBUTION PRIVILEGES for IrDA MEMBERS ONLY:

IrDA Members Limited Reproduction and Distribution Privilege: A limited privilege of reproduction and distribution of IrDA copyrighted publications is granted to IrDA members in good standing and for sole purpose of reasonable reproduction and distribution to non-IrDA members who are engaged by contract with an IrDA member for the development of IrDA certified products. Reproduction and distribution by the non-IrDA member is strictly prohibited.

TRANSACTION NOTICE to IrDA MEMBERS ONLY:

Each and every copy made for distribution under the limited reproduction and distribution privilege shall be conspicuously marked with the name of the IrDA member and the name of the receiving party. Upon reproduction for distribution, the distributing IrDA member shall promptly notify IrDA (in writing or by e-mail) of the identity of the receiving party.

A failure to comply with the notification requirement to IrDA shall render the reproduction and distribution unauthorized and IrDA may take appropriate action to enforce its copyright, including but not limited to, the termination of the limited reproduction and distribution privilege and IrDA membership of the non-complying member.

TRADEMARKS:

1. Prohibitions: IrDA claims exclusive rights in its trade names, trademarks, service marks, collective membership marks and certification marks (hereinafter collectively "trademarks"), including but not limited to the following trademarks: INFRARED DATA ASSOCIATION (wordmark alone and with IR logo), IrDA (acronym mark alone and with IR logo), IR logo, IR DATA CERTIFIED (composite mark), and MEMBER IrDA (wordmark alone and with IR logo). Any unauthorized use of IrDA trademarks is strictly prohibited.

2. Authorized Use: Any authorized use of a IrDA collective membership mark or certification mark is by NONEXCLUSIVE USE LICENSE ONLY. No rights to sublicense, assign or transfer the license are granted and any attempt to do so is void.

NO REPRESENTATION of THIRD PARTY RIGHTS:

IrDA makes no representation or warranty whatsoever with regard to IrDA member or third party ownership, licensing or infringement/non-infringement of intellectual property rights. Each recipient of IrDA publications, whether or not an IrDA member, should seek the independent advice of legal counsel with regard to any possible violation of third party rights arising out of the use, attempted use, reproduction, distribution or public display of IrDA publications.

IrDA assumes no obligation or responsibility whatsoever to advise its members or non-members who receive or are about to receive IrDA publications of the chance of infringement or violation of any right of an IrDA member or third party arising out of the use, attempted use, reproduction, distribution or display of IrDA publications.

LIMITATION of LIABILITY:

BY ANY ACTUAL OR ATTEMPTED USE, REPRODUCTION, DISTRIBUTION OR PUBLIC DISPLAY OF ANY IrDA PUBLICATION, ANY PARTICIPANT IN SUCH REAL OR ATTEMPTED ACTS, WHETHER OR NOT A MEMBER OF IrDA, AGREES TO ASSUME ANY AND ALL RISK ASSOCIATED WITH SUCH ACTS, INCLUDING BUT NOT LIMITED TO LOST PROFITS, LOST SAVINGS, OR OTHER CONSEQUENTIAL, SPECIAL, INCIDENTAL OR PUNITIVE DAMAGES. IrDA SHALL HAVE NO LIABILITY WHATSOEVER FOR SUCH ACTS NOR FOR THE CONTENT, ACCURACY OR LEVEL OF ISSUE OF AN IrDA PUBLICATION.

DISCLAIMER of WARRANTY:

All IrDA publications are provided "AS IS" and without warranty of any kind. IrDA (and each of its members, wholly and collectively, hereinafter "IrDA") EXPRESSLY DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE AND WARRANTY OF NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS. IrDA DOES NOT WARRANT THAT ITS PUBLICATIONS WILL MEET YOUR REQUIREMENTS OR THAT ANY USE OF A PUBLICATION WILL BE UN-INTERRUPTED OR ERROR FREE, OR THAT DEFECTS WILL BE CORRECTED. FURTHERMORE, IrDA DOES NOT WARRANT OR MAKE ANY REPRESENTATIONS REGARDING USE OR THE RESULTS OR THE USE OF IrDA PUBLICATIONS IN TERMS OF THEIR CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE. NO ORAL OR WRITTEN PUBLICATION OR ADVICE OF A REPRESENTATIVE (OR MEMBER) OF IrDA SHALL CREATE A WARRANTY OR IN ANY WAY INCREASE THE SCOPE OF THIS WARRANTY.

LIMITED MEDIA WARRANTY:

IrDA warrants ONLY the media upon which any publication is recorded to be free from defects in materials and workmanship under normal use for a period of ninety (90) days from the date of distribution as evidenced by the distribution records of IrDA. IrDA's entire liability and recipient's exclusive remedy will be replacement of the media not meeting this limited warranty and which is returned to IrDA. IrDA shall have no responsibility to replace media damaged by accident, abuse or misapplication. ANY IMPLIED WARRANTIES ON THE MEDIA, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO NINETY (90) DAYS FROM THE DATE OF DELIVERY. THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM PLACE TO PLACE.

CERTIFICATION and GENERAL:

Membership in IrDA or use of IrDA publications does NOT constitute IrDA certification of products. It is the sole responsibility of each manufacturer, whether or not an IrDA member, to obtain certification of products in accordance with IrDA rules for certification.

All rights, prohibitions of right, agreements and terms and conditions regarding use of IrDA publications and IrDA rules for certification of products are governed by the laws and regulations of the United States. However, each manufacturer is solely responsible for compliance with the import/export laws of the countries in which they conduct business. The information contained in this document is provided as is and is subject to change without notice.

Contents

1. INTRODUCTION **1**

1.1 Overview **1**
1.2 Device Types **1**
1.3 Terminology **2**
1.4 Byte Ordering **3**
1.5 References **4**

2. IRCOMM SERVICE TYPE OVERVIEW **5**

2.1 3-Wire Raw **5**
2.1.1 IrLPT 5
2.2 3-Wire **6**
2.3 9-Wire **6**
2.4 Centronics **6**
2.5 Summary **7**

3. SERVICE INTERFACE DEFINITION **8**

3.1 Service Definition Model **8**
3.2 Connect services **9**
3.3 Disconnect service **10**
3.4 Data service **11**
3.5 Control service **12**

4. FLOW CONTROL **14**

4.1 Tiny TP and IrLAP flow controls in overview **14**
4.2 Wired serial port flow control **14**
4.2.1 XON/XOFF 14
4.2.2 ENQ/ACK 14
4.2.3 Hardware flow control 14
4.3 Port Emulation Entity serial flow control **14**
4.3.1 XON/XOFF 15
4.3.2 ENQ/ACK 15
4.3.3 RTS/CTS 16
4.3.4 DTR/DSR 16
4.4 IrCOMM Centronics flow control **16**

5. FRAME FORMATS AND THE CONTROL CHANNEL **17**

5.1 Frame Formats **17**
5.2 General Control Parameters **18**
5.2.1 Service Type 19
5.3 When to send the general control parameter **19**
5.4 Data Channel Format **19**

6. DISCOVERY AND IRLMP IAS OBJECTS **21**

6.1 IrCOMM hint bits **21**
6.2 IrCOMM IAS entry **21**
6.2.1 LsapSel Attribute 22
6.2.2 Parameters Attribute 22
6.2.3 InstanceName Attribute 24
6.3 Advertising multiple service types **24**
6.4 IrLPT IAS Entry **25**

7. STATE DEFINITION AND TRANSITIONS **26**

7.1 Start Chart **26**
7.2 State Definitions **26**
7.3 Event Descriptions **27**
7.4 Action Descriptions **27**

8. 3-WIRE RAW AND IRLPT IN DETAIL **31**

8.1 How 3-Wire raw and IrLPT differ **31**
8.2 IAS entry and hint bits **31**
8.3 Basic link operation **31**
8.4 Handling the non-data circuits **32**

9. 3-WIRE IN DETAIL **33**

9.1 IAS entry and hint bits **33**
9.2 Basic link operation **33**
9.3 Control channel usage in 3-Wire **33**
9.3.1 Data Rate 35
9.3.2 Data Format 35
9.3.3 Flow Control 35
9.3.4 XON/XOFF and ENQ/ACK Flow Control Characters 35
9.3.5 Line Status 35
9.3.6 Break 35
9.4 Parameters Sent at Connection Time **35**
9.5 Parameters Sent During a Connection **36**
9.6 Handling the non-data circuits **36**

10. 9-WIRE IN DETAIL **37**

10.1 IAS and Hint Bits **37**
10.2 Basic link operation **37**
10.3 Control Parameters **37**
10.3.1 DTE Line Settings and Changes 38
10.3.2 DCE Line Settings and Changes 38
10.3.3 Poll for Line Settings 38
10.4 Parameters Sent at Connection Time **38**
10.5 Parameters Sent During a Connection **39**
10.6 Null Modem Emulation **39**

11. CENTRONICS IN DETAIL **41**

11.1 IAS and Hint Bits **41**
11.2 Basic link operation **41**
11.2.1 Traditional or compatible parallel interface emulation 41
11.2.2 IEEE 1284 emulation 41
11.3 Centronics control channel parameters **41**
11.3.1 Status Query 44
11.3.2 Set Busy Time-out 44
11.3.3 IEEE 1284 Mode Support 44
11.3.4 IEEE 1284 Device ID 44
11.3.5 Select IEEE 1284 Mode 44
11.3.6 IEEE 1284 ECP/EPP data transfer 44

12. ANNEX A IR TERMINAL ADAPTER (IRTA) **46**

12.1 Model and components **46**
12.1.1 IR-DTE 47
12.1.2 IrTA 48
12.1.3 Interface 49
12.2 IrTA specific requirements **50**

12.2.1 Requirements for Port Emulation Entity in IR-DTE 50
12.2.2 Requirements for IrTA 50
12.2.3 Requirements for DCE 51
12.3 Service Definition 51
12.3.1 Service Elements Between Port Emulation Entity and IrLMP in the IR-DTE 52
12.3.2 Service Elements Between Port Emulation Entity and IrCOMM in the IR-DTE 52
12.3.3 Service Elements Between IrTA and IrLMP in the IrTA device 52
12.3.4 Service Elements Between IrTA and IrCOMM in the IrTA device 53
12.3.5 Service Elements Between IrTA and DCE 53
12.4 State Transition Description of IrTA 56
12.4.1 General Description 57
12.4.2 Status Machine Rules 57
12.4.3 IrTA State Transition Table 57
12.4.4 State Definitions 64
12.4.5 State Variables 64
12.4.6 Event Descriptions 64
12.4.7 Action Descriptions 65
12.5 IrTA Service Sequence Example 69
12.5.1 Normal (Call Connection Phase) 69
12.5.2 Normal(Data Transfer Phase; Flow control:RTS/CTS control) 69
12.5.3 Normal(Data Transfer Phase; Flow control:XON/XOFF control) 70
12.5.4 Normal (Call Disconnection Phase) 71
12.5.5 Abnormal(Enforced Disconnection from IR-DTE) 71
12.5.6 Abnormal(Abnormal Disconnection from IR Link) 71
12.5.7 Abnormal(Abnormal Disconnection from IrTA) 72
12.5.8 Abnormal(Enforced Disconnection from DCE) 72
12.5.9 Abnormal (Enforced Disconnection Partner Terminal or Network) 72
12.6 Implementation alternative of IrTA and IR-DTE 72
12.6.1 IrTA procedure in the disconnection request from Network (PSTN/ISDN) via DCE 73
12.6.2 IrTA procedure when the IrCOMM disconnected during the connection phase between DCE and network (PSTN/ISDN) 73
12.6.3 Start of the establishment of the IrCOMM link 74
12.6.4 Treatment of Break signal 76

13. APPENDIX A. TYPICAL PROTOCOL SEQUENCE EXAMPLES 77

13.1 3-Wire raw 77
13.2 3-Wire, Service type is sent by TTP_Connect 78
13.3 9-Wire, Service type is sent by TTP_Data 79

14. APPENDIX B INTERFACING IRCOMM TO DATA OR FAX MODEM 80

14.1 Naming, References to External Entities 80
14.2 External Interfaces 80
14.3 Flow Control 81
14.3.1 XON/XOFF flow control 82
14.3.2 RTS/CTS flow control 82
14.3.3 DSR/DTR flow control 83
14.4 Procedures for Changing Communication Settings 85
14.5 Internal Organization of IrCOMM 86
14.6 IrCOMM Client Control 87
14.6.1 Purpose 87
14.6.2 Overview 87
14.6.3 IrCOMM Client Control State Transition Diagram 87
14.6.4 IrCOMM Client Control State Transition Table 87
14.7 IrCOMM Host Control 88
14.7.1 Purpose 88
14.7.2 Overview 88

14.7.3 IrCOMM Host Control State Transition Diagram 88
14.7.4 IrCOMM Host Control State Transition Table 89
14.8 DCE Initiated Connection Establishment - Incoming Call 89
14.9 Requirements 89
14.10 Functionality 90

1. Introduction

This document defines IrCOMM, the emulation of Serial and Parallel ports over the IrLMP/IrLAP protocol stack. The motivation for IrCOMM comes from the many printing and communication applications which use standard communication APIs to talk to other devices via serial and parallel ports. By making IrDA protocols accessible via these APIs, many existing applications including printing can run over an IrDA infrared link without change. This intent to support so-called legacy applications is the basis for IrCOMM. New applications are encouraged to take better advantage of IrDA protocols by using their capabilities directly.

1.1 Overview

Emulating COMM ports raises a number of questions, starting with what kinds of ports will be emulated. IrCOMM emulates RS-232 (EIA/TIA-232-E) serial ports, and Centronics parallel ports like those found on most personal computers. The four service types used to emulate these ports are the core of this specification. Before discussing service types, however, there are some basic differences between wired and IrDA communications to consider.

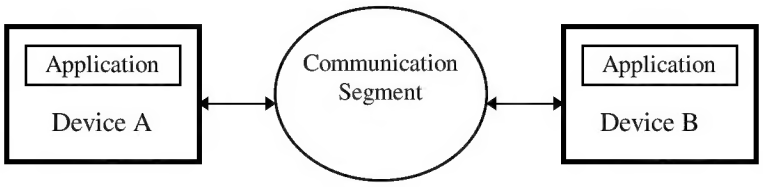
Wired communications methods can send **streams of information in both directions at once,** because there are multiple wires (some to send data on, some to receive data from). With infrared, there is the equivalent of only one wire (the IR path through the air), which has the following implications:

- IrDA protocols send **packets one way at a time**. If a device tried to send data and listen for data at the same time, it would "hear" itself and not the device it wants to communicate with. The way IrDA devices achieve two way communications is to take turns, also known as "turning the link around". This happens at least every 500 milliseconds, and can be made more frequent as necessary. This latency makes it impossible to perfectly emulate the wired COMM environment - very timing sensitive operations will be disrupted. Fortunately, many communication tasks are not so sensitive, and these can use IrCOMM.
- All of the information carried on multiple wires must be carried on the single IR "wire". This is accomplished by subdividing the packets into data and control parts. In this way a logical data channel and control channel are created, and the various wires can be emulated.

On a different level, IrCOMM is intended for legacy applications, applications that know about serial or parallel ports but know nothing about IrDA protocols. IrDA protocols, however, have very different procedures and APIs from wired COMMs. Suppose, for example, a word processing application wants to print via IR using IrDA protocols, an application must first "discover" the printer (locate a printer in IR-space), then check the printer's IAS to find information needed to connect. Since the word processing application (a legacy application) knows nothing about this, IrCOMM maps these operations into normal COMM operations so that it is completely transparent.

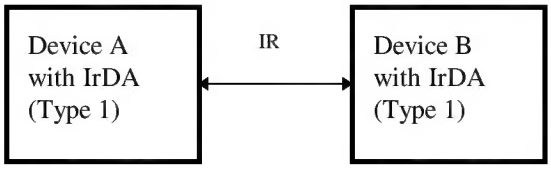
1.2 Device Types

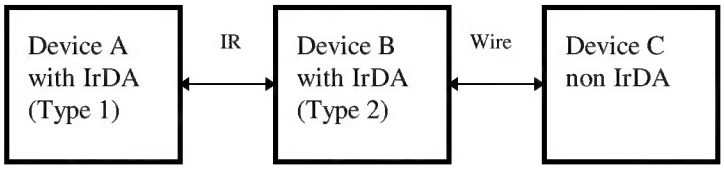
For the purposes of IrCOMM a complete communication path involves two applications running on different devices (the communication endpoints) with a communication segment between them. The communication segment may consist solely of IR or IR connections to a network. The figure below shows the complete communication path.



IrCOMM is intended to cover applications that make use of the serial and parallel ports of the devices in which they reside. In the simple case, the communication segment is an IR link from one device to another (direct connect). In the case where the communication segment is a network, IR is used for the path between the device and a networking connection device like a modem. Modems communicate through the network using wire, radio or IR. IrCOMM is only concerned with the connection between devices in the direct connect case or between the device and a modem in the network case. There are other configurations that IrCOMM can support like modules that communicate via IR on one side and provide a wired interface on the other side. These devices are not really modems but offer a similar service and thus, are not explicitly discussed.

Basically two device types exist that IrCOMM must accommodate. Type 1 devices are the communication end points like computers and printers. Type 2 devices are those that are part of the communication segment like modems. Though IrCOMM does not make a distinction between these two device types in the protocol, accommodating both types of devices impacts the IrCOMM protocol. The figures below illustrate how the two IrCOMM device types fit into communication paths.





The information transferred between two IrCOMM entities has been defined to support both type 1 and type 2 devices. Some information is only needed by type 2 devices while other information is intended to be used by both. In the protocol no distinction is made between type 1 and type 2 therefore, it is up to the IrCOMM implementor to determine if the information passed in the IrCOMM protocol is of use to the implementation. Since the devices do not know the type of the other device in the communication path they must pass all the information specified by the protocol of which they have knowledge.

1.3 Terminology

The following terms are used throughout the document.

Computer This term is used to cover all computing devices like PCs (IBM Compatibles, MACs, etc.), Workstations, PDAs, Palmtops, Electronic Organizers, etc.

Peripherals This term is used to cover devices like printers, modems and other devices that are

traditionally connected to Computers via Serial and/or Parallel cables.

Client/Server The participants in an IrCOMM communication session are divided into clients and servers. **Clients** are the devices or applications that initiate the communication. In the IrDA world they perform the discovery, query the IAS, and initiate the connection. **Servers** are the devices and applications to which **clients** connect. Printers and modems are typical **servers**. When talking to printers and modems, computers are **clients**.

LPT/COM **LPT** denotes a parallel port. **COM** denotes a serial port.

DTE Data Terminal Equipment - in serial communications, DTE refers to a device at the endpoint of the communications path; typically a computer or terminal of some kind.

DCE Data Circuit-Terminating Equipment - in serial communications, DCE refers to a device between the communication endpoints whose sole task is to facilitate the communications process; typically a modem.

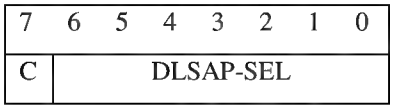
User data It is convenient to have a term that refers to the data that in the serial case travels over TD and RD, or in the parallel case travels over the 8 data lines. The term **user data** is used for this purpose.

Control data (or control information) All the data other than the user data. For the most part, information that travels on the non-data circuits of the serial and parallel ports.

1.4 Byte Ordering

This document represents frames as collections of bytes (octets) with each byte being composed of 8 bits numbered 0-7. Bit 0 is always the least significant bit (LSB) and bit 7 is always the most significant bit (MSB). Bytes are represented throughout this document in the following forms:

- Diagrammatic - a byte is represented as a rectangle. In some cases bit fields have special meaning and are indicated for clarity. The most significant bit is the bit on the left and the least significant bit is the bit on the right. An example is given below



- Hexadecimal - a byte is represented with two hex digits with the least signification nibble on the right, the most significant nibble on the left, and both digits preceded by 0x. An example is the value 5 which is written as 0x05.
- Tabular - a byte is represented by a table with each row of the table corresponding to a bit. The least significant bit occupies the first row of the table and the most significant bit occupies the last row of the table. An example is given below.

Byte 2	
Bit	Function
8	Telephony
9	File Server
10	IrCOMM
11	reserved
12	reserved
13	reserved
14	reserved
15	Extension

1.5 References

[IRDAIRLAP] Infrared Data Association, "Serial Infrared Link Access Protocol (IrLAP)", Version 1.0

[IRDAIRLMP] Infrared Data Association, "Serial Infrared Link Management Protocol (IrLMP)", Version 1.0

[IRDATINYTP] Infrared Data Association, "'Tiny TP': A Flow-Control Mechanism for use with IrLMP", Version 1.0

[TIA232] EIA/TIA-232-E (July 1991), "Interface between Data Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange".

[ITU-TV24] ITU-T V.24 (BB 8.1, 1988), "List of Definitions for Interchange Circuits between Data Terminal Equipment (DTE) and Data Circuit-Terminating Equipment (DCE)"

[IEEE1284] IEEE Std 1284-1994, "IEEE Standard Signaling Method for a Bi-directional Parallel Peripheral Interface for Personal Computers".

[ITU-TV.110] ITU-T Recommendation V.110 "Support of data terminal equipment (DTEs) with V-series type interfaces by an integrated services digital network (ISDN) "

[ITU-TV.42] ITU-T Recommendation V.42 "Error-Correcting Procedures for DCEs Using Asynchronous-to-synchronous Conversion"

2. IrCOMM Service Type Overview

IrCOMM emulates serial and parallel ports. However, printing and communications applications use communication ports in a variety of ways. To address this need, IrCOMM provides four service types or classes: 3-Wire raw, 3-Wire, 9-Wire, and Centronics. The service types fall into 2 camps, which are called raw and cooked; the differences hinge on whether a control channel is supplied and the type of flow control used. 3-Wire raw provides a data channel only, and uses IrLAP flow control. The "cooked" service types (3-Wire, 9-Wire, and Centronics) support a control channel, and employ Tiny TP flow control as described in [IRDATINYTP]. The meaning, format, and use of the control channel is explained in a later chapter.

2.1 3-Wire Raw

The 3-Wire raw service type can be used for serial or parallel emulation when a single exclusive connection is acceptable, and only the data circuits need to be emulated (no non-data circuits in a corresponding wired setting carry any information). The name of this service comes from the notion of emulating the minimum three RS-232 circuits (see Ref. [ITU-TV24]) required for full duplex communications. The circuits are shown below.

102	Signal Common	This circuit is not needed for IR but is shown because it is one of the circuits that drove the definition of the name.
103	Transmitted Data (TD)	This circuit carries data transmitted by the DTE
104	Received Data (RD)	This circuit carries data received by the DTE

Here are the main attributes of the 3-Wire Raw service class:

- **Only one non-IAS IrLMP connection can be open if 3-Wire raw is used** - all other connections must be closed before it can be established, and others must wait until the raw connection is closed before they can connect. This is because 3-Wire raw uses the flow control features of IrLAP, which can result in a deadlock condition if more than one non-IAS connection is open.
- Minimal Implementation. All the IrCOMM data is sent directly over IrLMP in IrLMP packets. All data that follows the IrLMP Mux bytes in an IrLMP packet is IrCOMM **data,** (i.e. it is the information that would travel over the data line(s) of a wired interface). No control channel is available to communicate information about the state of other leads (e.g. RTS/CTS), software flow control settings, and the like. A service which employs 3-Wire raw must be able to do without that information. The link is merely a raw channel for the movement of data.
- This service can be used to emulate both serial and parallel ports. This may seem counter-intuitive (who has ever heard of a 3-Wire parallel port?), but if you remove the non-data circuits (which 3-Wire raw does not emulate), serial and parallel are equivalent - just streams of data.

2.1.1 IrLPT

IrLPT is an IrDA service in use on commercially available printing devices. It is equivalent to 3-Wire raw in functionality, but is slightly different in how it uses the IAS. See the chapter called "3-Wire Raw and IrLPT in Detail" for more information.

2.2 3-Wire

Like 3-Wire raw, the name of this service comes from the minimum three RS-232 circuits required for full duplex communications. Like 3-Wire raw, it is intended for both serial and parallel ports. However, there are the following important differences:

- 3-Wire service class makes use of Tiny TP flow control, so that it may coexist with other connections that employ higher level (not IrLAP) flow control (including other cooked IrCOMM connections). It is not limited like 3-Wire raw to a single IrLMP connection.
- 3-Wire service class supports a control channel for sending information like data format. The control channel mechanism is described in the chapter titled Frame Formats and the Control Channel.
- Because of the need for flow control and the use of the control channel, the 3-Wire service type uses a more elaborate frame format.

2.3 9-Wire

The name of this service class comes from the notion of emulating the 9 circuits of an RS-232 interface which are part of a standard IBM compatible PC. Unlike the previous services it is true to its name; 9-Wire emulates serial ports only. Three of the circuits are the same as described in the 3-Wire service classes. The other six are listed below

105	Request to Send (RTS)
106	Clear to Send (CTS)
107	Data Set Ready (DSR)
108/2	Data Terminal Ready (DTR)
109	Data Channel Received line signal detector (RLSD), aka Carrier Detect (CD)
125	Calling indicator, aka Ring Indicator (RI)

Some attributes of this service are list below.

- Like 3-Wire, it uses the Tiny TP flow control mechanism. It also uses the same control channel mechanism for sending information like data format.
- The control channel is used to send the states of the other RS-232 leads as they change.

2.4 Centronics

This service is intended to emulate the function of a standard Centronics interface. This service is for parallel ports only. Some attributes of this service are listed below.

- It uses the Tiny TP flow control mechanism.
- It uses the same control channel mechanism used in 3-Wire to send the status/changes of the additional circuits.

See the chapter Centronics in Detail for information on the circuits emulated by Centronics.

2.5 Summary

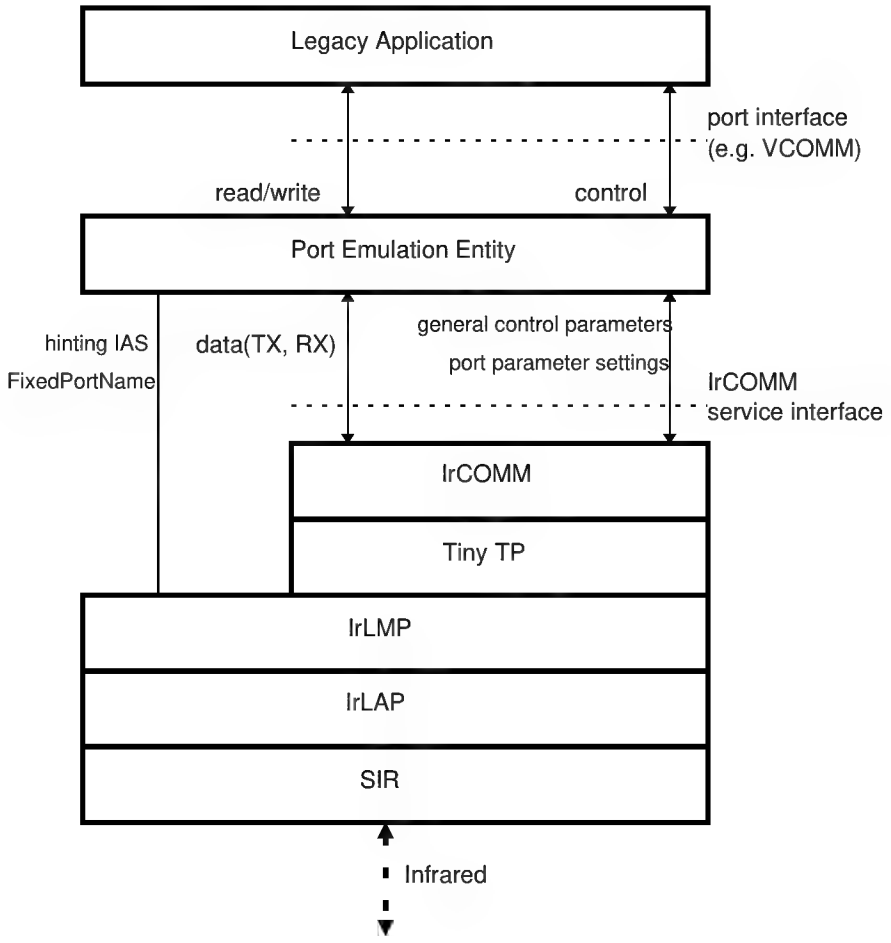
3-Wire raw, 3-Wire, 9-Wire, and Centronics make up the four service types defined by IrCOMM. In theory 9-Wire and Centronics service types would cover all the connections anyone needs, but the combination of historical offerings and the desire for minimal implementations has driven the specification of four service types. The next few chapters discuss the issues of service interface, flow control, control channel parameters, and discovery/IAS services. The document concludes with a detailed specification of each service type.

3. Service Interface Definition

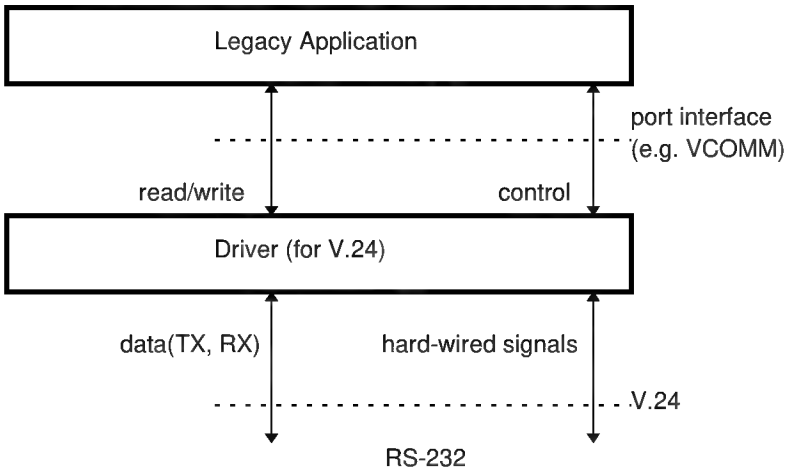
IrCOMM is intended to define a protocol that can be used to emulate serial and parallel ports. In most systems IrCOMM will be part of a port driver which includes a port emulation entity that must support an existing communication API. The communication APIs vary from operating system to operating system and device to device. This document does not specify how IrCOMM is used by the port driver to emulate an existing API but instead focuses on a set of services that can be used by all port drivers. Port drivers are not required to utilize all the services of IrCOMM. In fact it is the job of the port driver implementor to properly map the services of IrCOMM to the particular system.

3.1 Service Definition Model

The figure below shows a model of how IrCOMM fits into a typical system. This figure represents the IrCOMM reference model.



The figure below shows the wired version that is being emulated by the IrCOMM reference model



The elements for the IrCOMM reference model are described below

Legacy Application Applications which utilize conventional serial/parallel port communication interface.

Port Emulation Entity The port emulation entity maps a system specific communication interface (API) to the IrCOMM services. It is also responsible for device discovery and LM_IAS queries. The port emulation entity plus IrCOMM make up a port driver.

IrCOMM Provides a transparent data stream channel and control channel over an IrLMP link or Tiny TP link.

TinyTP Provides a data stream channel to IrCOMM along with a flow control mechanism.

IrLMP, IrLAP, SIR Link protocols defined by IrDA

Port interface (e.g. VCOMM) Application programmer's interface (API) for communication. This interface varies from system to system. An example is the VCOMM interface of Windows95.

IrCOMM service interface The IrCOMM interface provides the following services.

- Connect
- Disconnect
- Data
- Control

3.2 Connect services

IrCOMM_Connect.req(CalledLsap, ServiceType, InitialControlParameters, QoS)

IrCOMM_Connect.ind(CallingLsap, ServiceType, InitialControlParameters, QoS)

IrCOMM_Connect.rsp(InitialControlParameters)

IrCOMM_Connect.cnf(InitialControlParameters)

The Connect services are used to establish an IR link with a peer IrCOMM system. The connection services are a confirmation type service. Upon receipt of an IrCOMM_Connect.ind primitive the responding Port Emulation Entity (IrCOMM user) must either accept or reject the incoming connection. Connections are accepted by an invocation of IrCOMM_Connect.rsp or are rejected by an invocation of IrCOMM_Disconnect.req with a reason of 'User Disconnect'.

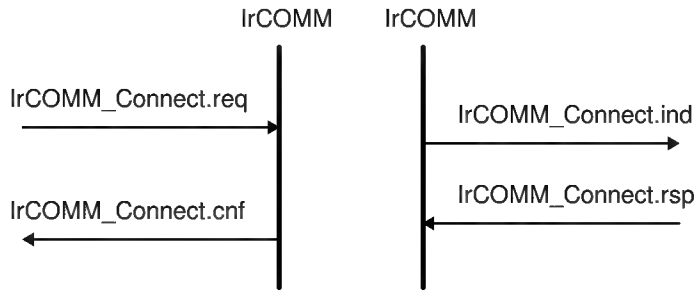
Parameters used in this definition are as follows.

CallingLsap, CalledLsap Caller and callee Lsap address respectively. 'CalledLsap' is discovered by the Port Emulation Entity using IrLMP discovery and IAS facilities, however, the exact algorithms used are out of scope of this document.

ServiceType Service type of IrCOMM emulation. It should be one of '3-Wire raw', '3-Wire', '9-Wire', or 'Centronics'. The value 'Default' is also permitted. Usually the initiating entity will query the IAS of the responding entity for the service types supported by the responder and use one of those.

InitialControlParameters is an optional parameter and is the initial set of values assignments for the port communication settings and line settings. The InitialControlParameters are limited to a total encoded size of 60 octets.

QoS Quality of service parameter used for IrLAP link. 'QoS' includes data rate, maximum turn around time, data size, and disconnection threshold. It is implementation specific whether this parameter is really reflected.



3.3 Disconnect service

IrCOMM_Disconnect.req(UserData)

IrCOMM_Disconnect.ind(Reason, UserData)

The Disconnect service is used to end the connection between IrCOMM entities. The user of IrCOMM is always permitted to use this service whenever it wishes to release the connection. The Disconnect service is used in these cases.

- If a Port Emulation Entity (IrCOMM user) wishes to release or abort an IrCOMM connection with a peer IrCOMM entity, it will use this service.
- If the underlying IR connection is disconnected, IrCOMM will notify the Port Emulation Entity via an IrCOMM_Disconnect.ind.
- A Port Emulation Entity uses Disconnect service to refuse an incoming connection.

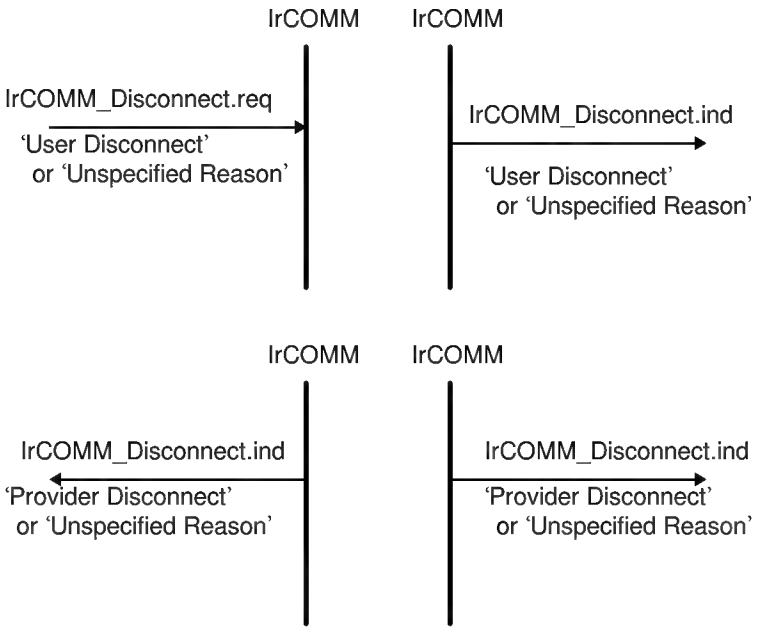
- A Disconnect.ind is issued if the underlying layer failed to establish a connection.

Parameter used in the Disconnect services are as follows.

UserData UserData is any octet string up to 60 octets and is optional.

Reason This parameter indicates the reason why a link is disconnected or why a connection is refused. This parameter is optional. If used, 'Reason' should be one of the following:

User Disconnect	This value is used when the responder refuse to make a IrCOMM connection, and when IrCOMM user wishes to disconnect the existing connection.
Provider Disconnect	This value is used when the provider of IrCOMM connection (IrCOMM or underlying protocol stack) causes a disconnection.
UnSpecified Reason	This value is used when the reason is unspecified in this document.



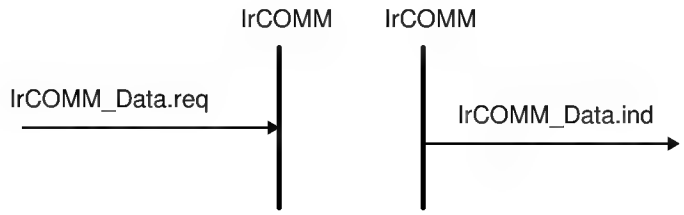
3.4 Data service

IrCOMM_Data.req(Data)

IrCOMM_Data.ind(Data)

The Data service is used to convey data between Port Emulation Entities (IrCOMM users). Each item of data (single octet or multiple octets) is regarded as part of a contiguous stream of data. IrCOMM_Data service is for a reliable data transfer. The Parameter used in this definition is as follows.

Data Data of Port Emulation Entity to be sent.



3.5 Control service

IrCOMM_Control.req(ControlParameters)

IrCOMM_Control.ind(ControlParameters)

The Control service is used to convey control parameters between Port Emulation Entities (IrCOMM users). Default values should be assumed if no control parameters has been designated since the connection has been made. The parameter used in this service is as follows.

ControlParameters Sequence of Control Parameters. Control parameters are described in detail later in this document. Below is a list of parameters based on service type.

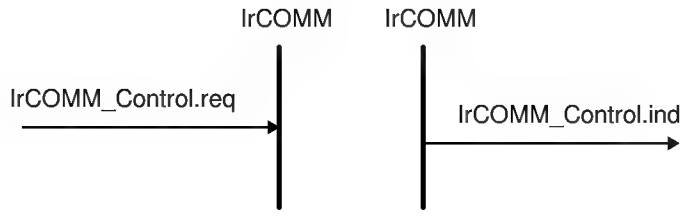
(1) General control parameters
- Service Type

(2) Control parameters for 3-Wire and 9-Wire service type
- Data rate
- Data Format
- Flow control
- XON/XOFF flow control characters
- ENQ/ACK flow control characters
- Line status
- Break

(3) Control parameters for 9-Wire service type
- DTE Line Settings and Changes
- DCE Line Settings and Changes
- Poll for Line Settings

(4) Control parameters for Centronics service type
- Status query
- Set Busy Timeout
- Request IEEE 1284 Mode Support
- Request IEEE 1284 Device ID
- Select IEEE 1284 Mode

- IEEE 1284 ECP/EPP data transfer
- Status query response
- Set Busy Timeout response
- IEEE 1284 Mode Support response
- IEEE 1284 Device ID response
- Select IEEE 1284 Mode response
- IEEE 1284 ECP/EPP data transfer response



4. Flow Control

Wired ports commonly use flow control mechanisms such as XON/XOFF to control communications. On the other hand, all IrCOMM services use either Tiny TP or IrLAP flow control on the infrared link. This section describes how these two different approaches are managed in IrCOMM.

4.1 Tiny TP and IrLAP flow controls in overview

3-Wire, 9-Wire and Centronics service classes are built upon a flow control mechanism called Tiny TP (see [IRDATINYTP]). Tiny TP offers both flow control, and segmentation and reassembly. IrCOMM uses only the flow control portion. Therefore, IrCOMM data must fit the negotiated IrLAP packet size (minus 4 bytes of overhead for IrLMP, Tiny TP, and the control channel header). This has no impact for user data, since IrCOMM just takes a stream of bytes from one side, carries it across in packets, and streams it up to the receiving API. However, control parameters are multi-byte entities, and must fit entirely within a single IrLAP packet since they will not be reassembled on the receiving side.

In contrast to the cooked service types, 3-Wire Raw uses IrLAP flow control and thus does not carry the (minimal) overhead of Tiny TP. Nor does it support the control channel, so 3-Wire raw does not have the control parameter/packet size issue described for the cooked types. As mentioned numerous times, however, a device can only have one non-IAS connection open if 3-Wire raw is in use.

4.2 Wired serial port flow control

Wired serial port flow controls fall into two camps - software flow control using characters such as XON/XOFF, and hardware flow control using the RTS/CTS and DTR/DSR circuits. These methods may be used by both sides of a wired link, or may be used only in one direction, depending on the device's needs (some devices are at more risk for running out of buffer space than others). Neither camp is particularly similar to the credit based scheme in Tiny TP. The following sections briefly summarize wired methods.

4.2.1 XON/XOFF

XON/XOFF flow control makes use of characters called XON and XOFF to control the link. If side A wants to flow control the link off, it sends XOFF across the wire to side B. Each side is always scanning incoming data for XON/XOFF, and when side B sees the XOFF, it stops sending data; when it later sees XON, it knows it can resume sending data.

4.2.2 ENQ/ACK

ENQ/ACK flow control makes use of characters called ENQ and ACK to control the link. If side A wants to make sure it does not overflow side B it sends an ENQ after sending *n* characters. It then waits until side B sends a ACK before sending any more data.

4.2.3 Hardware flow control

There are two main varieties of hardware flow control: RTS/CTS and DTR/DSR. In RTS/CTS flow control if the DCE wishes to flow control the link off it sets CTS low. When CTS goes low the DTE will stop sending characters. Setting CTS high will cause the DTE to resume sending. In the other direction, the DTE sets RTS low to cause the DCE to stop sending characters, and sets it high to resume.

DTR/DSR works very much like RTS/CTS. The DCE uses DSR to control data coming from the DTE and the DTE uses DTR to control the data coming from the DCE.

4.3 Port Emulation Entity serial flow control

On Type 1 devices some port drivers (Port Emulation Entities plus IrCOMM) will need to provide flow control services as specified by the API they are emulating. An application may request a particular flow control mechanism like XON/XOFF or RTS/CTS and expect the port driver to handle the flow control. On

Type 2 devices the port driver may need to perform flow control on the non-IrDA portion of the communication path (e.g. the flow control for a Type 2 device attached to a modem is between the device and the modem). This flow control is specified via the control parameters sent by the peer IrCOMM entity (usually a Type 1 device). The description of flow control in this section is for port drivers on Type 1 devices.

Since IrCOMM already has it's own flow control mechanism the port driver does not need to **perform** flow control using the methods requested by the application. In the ideal case, the application sets a flow control mechanism and assumes that the COMM system will handle the details. The port driver could simply ignore the request and rely on IrCOMM's flow control. The application is able to send and receive data, and does not know or care that the port driver did not perform flow control using the mechanism requested. However, in the real world some problems arise.

- The IrCOMM based port driver is running on top of a packet based protocol where data may be buffered somewhere in the communication path. Thus, the port driver cannot flow control off with the same precision as in the wired case.
- The application may decide to apply the flow control mechanism itself in addition to requesting flow control from the port driver.

These problems suggest that the port driver must do some additional work to perform flow control emulation properly. Here are the basic rules for flow control emulation.

- The port driver will not attempt to use the mechanism requested by the application but will rely on the flow control of IrCOMM.
- The port driver must be aware of the flow control mechanisms requested by the application and behave like the wired case when it sees changes on the non-data circuits (hardware flow control) or flow control characters in the incoming data (software flow control). For example, if XOFF and XON characters would have been stripped in the wired case they must be stripped by the IrCOMM based port driver.
- If the application sets a flow control mechanism via the port driver interface and then proceeds to invoke the mechanism on its own, the port driver must behave in a manner similar to that of the wired case (e.g. If XOFF and XON characters would have been passed through to the wire in the wired case the port driver must also pass these characters). In this case it is also possible for the port driver to perform local flow control with the application to increase the precision lost by buffered data, but this local behavior is beyond the scope of the IrCOMM specification.

These basic rules are applied to emulate each of the wired flow control schemes. Note that multiple types of flow control can be set at the same time. The following sections discuss the details of each flow control mechanism.

4.3.1 XON/XOFF

To perform flow control emulation of XON/XOFF, The port driver must scan the data received from the other side for the XON and XOFF characters. If these characters would have been consumed in the wired case they must be consumed by the IrCOMM based port driver.

As a purely local implementation issue, the port driver may need to scan the data coming from the local application. If it receives an XOFF or XON it must behave like the wired case, which in most systems is to transmit the character. However, it could also locally stop delivering data to the application when it sees an XOFF and resume delivering data when an XON is received. This would be done to account for the IrCOMM based port drivers larger amount of in-transit data in comparison to a wired connection.

4.3.2 ENQ/ACK

To perform flow control emulation of ENQ/ACK, the port driver must scan the data received from the other side and strip out ENQ and ACK characters. If these characters are passed to the port driver from the application they must be transmitted to the other side.

4.3.3 RTS/CTS

To perform flow control emulation of RTS/CTS, the port driver should initially indicate flow control ON by keeping the appropriate leads set high (DCE = CTS, DTE = RTS). If the application changes the state of the leads then the port driver must transmit these changes if it is emulating 9-Wire.

As a local implementation detail, the port driver could also stop delivering data to the application when the lead is set off and resume giving data to the application when the lead is set high.

4.3.4 DTR/DSR

To perform flow control emulation of DTR/DSR, the port driver should initially indicate flow control on by keeping the appropriate leads set high (DCE = DSR, DTE = DTR). If the application changes the state of the leads then the port driver must transmit this change if it is emulating 9-Wire.

As a local implementation detail, the port driver could also stop delivering data to the application when the lead is set off and resume giving data to the application when the lead is set high.

4.4 IrCOMM Centronics flow control

No attempt is made to simulate the hardware flow control lines used on an actual parallel port. Tiny TP flow control is sufficient for the parallel emulation in IrCOMM.

5. Frame Formats and the Control Channel

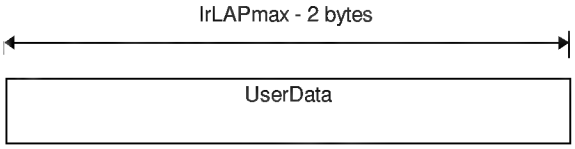
There are two data frame formats used in IrCOMM. A very simple format is used by 3-Wire raw; it is nothing more than user data (the bytes normally traveling on the data circuits). 3-Wire raw is said to have only a data channel. A more elaborate format is used by the cooked services to send additional information, creating in effect a second channel of information, called the control channel.

There are two uses for the control channel: to transmit the state of the non-data circuits in RS-232 and Centronics, and to transmit setup and status information between devices. For instance, a wired modem can determine baud rate, parity, and data size through analysis of the AT command. This is not possible in the IrCOMM case, so this information must be sent in the control channel. In another instance, a type 2 device (see figures in Chapter 1) may need to program a UART to properly deal with the wired device attached to it, so it needs the data format and flow control information sent over the control channel. The individual items carried by the control channel are called parameters.

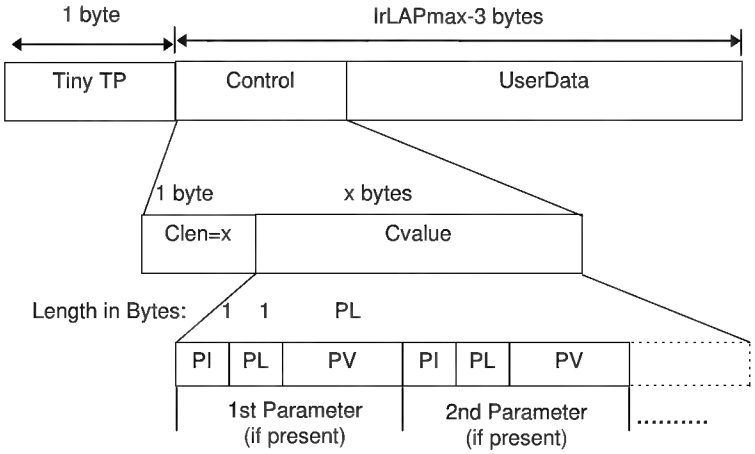
This section defines the general format of IrCOMM frames and control parameters, and the general control parameter used to select service type. Later chapters describe additional parameters, including data rate, flow control, line states and changes, error codes, and paper out indications.

5.1 Frame Formats

IrCOMM frames fit directly into the UserData field of IrLMP or TinyTP packets, which are based on the packet size that IrLAP computes (IrLAPmax) after negotiating the link. 3-Wire raw uses a very simple frame format, consisting of nothing but user data.



Now compare that with the data given to IrLMP for the cooked service cases:



The cooked service cases have user data, just as with raw, but precede the user data with two elements. The first element is one byte added by Tiny TP to carry credits to the other side and is not really part of the IrCOMM frame format. It should be noted that IrCOMM does not use the segmentation and reassembly (SAR) capabilities of TinyTP (MaxSduSize = 0). Thus, Connect and Connect Confirm TTP-PDUs do not contain a TTP parameters field (only the initial credit byte is present in these PDUs). The second element is the **control channel**, consisting of the control length byte and (optionally) some control data. The control

channel is truly part of the IrCOMM frame format. The control length byte holds the length (Clen) of the control data (Cvalue). The control data (if Clen > 0) immediately follows the control byte (Note: Clen is required even if there is no control data). All remaining bytes, if any, are user data. This mechanism allows control and data to exist in the same packet.

When a packet arrives that has both control parameters and data, **the control data will be processed and acted upon first, then the data will be processed.** It is permitted for a packet to have zero bytes of user data, or zero bytes of control data (Clen = 0). The control parameters are placed into the packet the order in which they occurred in time. The receiver of control parameters must process the parameters in the order they appear in the packet.

The control data (Cvalue) is composed of a sequence of **control parameters,** where each parameter is defined by three fields, PI, PL, PV. This is identical to the 3-tuple format for parameters used in IrLAP negotiation. PI is the parameter identifier or tag and is one byte in length. PL is the length of the parameter value and is one byte in length. PV is the parameter value and is PL bytes in length. Unless a parameter is defined with a PL equal to zero, parameters with PL equal to zero are considered improperly formed and should not be used. Beware of confusing control parameters with IAS parameters (defined in the next chapter). While similar in appearance, they are used at entirely different times, for different purposes, and are not interchangeable.

If the control channel information does not fit in a single frame, IrCOMM must send it in multiple frames. However, IrCOMM must assure that any single control parameter is NOT broken across frame boundaries (recall that the segmentation and reassembly capabilities of Tiny TP are not being used). Since the minimum IrLAP frame size is 64 bytes, and four of those are used already (two for IrLMP, one for Tiny TP, and one for the control length byte), the maximum size of a single control parameter is 60 bytes (1 PI + 1 PL + 58 PV). This applies to all control channel information including control information sent in Connect and Connect Confirm PDUs (Note that Connect LM-PDUs contain 2 extra bytes compared to Data LM-PDUs and thus, Connect and Connect confirm TTP-PDUs are not able to hold a maximum size IrCOMM parameter).

5.2 General Control Parameters

The control channel is used in all the cooked service types to carry control parameters (defined above). Some parameters defined later in this specification are used by just one or two of the service types. A general control parameter, however, is one which make sense for all cooked service types. The following table defines the single **general** control parameter, and also shows id ranges for the other control parameters.

PI	PI name	PL	PV datatype		PV Description	PV Default value, notes
0x00	Service Type	1	byte (bitmask)	bit 0	unused	highest order bit set in the IAS service type parameter
				bit 1	3-Wire	
				bit 2	9-Wire	
				bit 3	Centronics	
0x01-0x0F	Reserved for future general control ids					
0x10-0x1F	Reserved for 3-Wire ids					
0x20-0x2F	Reserved for 9-Wire ids					
0x30-0x3F	Reserved for Centronics ids					
0x40-0xFF	Reserved for future use					

Future versions of this specification may expand on the list of general parameters. See the 3-Wire, 9-Wire and Centronics chapters for definitions of other parameters. A brief description of the service type parameter is given below.

5.2.1 Service Type

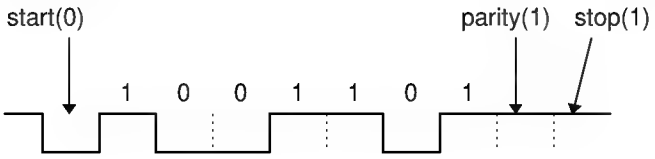
The service type control parameter is sent by the client (the IrCOMM entity originating the connection) to select the service type to be used during the connection. During discovery the client finds what service types are available from the service type parameter in the IAS Parameter attribute of the server. The client chooses one and communicates that choice with this control parameter. This parameter is for both Type 1 and Type 2 devices that offer more than one cooked service type. If not sent, the service type defaults to the highest of service types offered (Centronics, then 9-Wire, then 3-Wire).

5.3 When to send the general control parameter

The general control parameter described here is sent immediately upon establishment of the IrLMP connection, before any COMM data or other parameters are sent. It may be sent as the data portion of the TTP Connect Request or in a separate TTP data packet. If it is not sent, the default value is assumed to hold. Once sent, it may not be changed for the duration of the connection. The timing of other service-type-specific parameters is covered in the detailed descriptions of those service types.

5.4 Data Channel Format

The IrCOMM data channel provides a logical data stream for sending and receiving application data over a TinyTP or IrLAP connection. The data channel is octet (8 bits) aligned because the data transfer service provided by the IrDA protocols is based on packets of octets. In the case of serial emulation the data channel is emulating an asynchronous data stream of characters. The format of an asynchronous serial character is shown below.



There are many ways to encode the information of an asynchronous serial character in an octet of the data channel. In order to prevent ambiguous interpretation, these rules should be followed.

- The port parameter settings of data rate, data length, parity and stop bit are received from the other side via the control channel. This information should be saved while the port is active.
- No start bit and stop bit(s) are carried with the data bits.
- The parity information is not sent with the data but can be generated from the port parameters sent in the control channel.
- If the number of data bits is less than 8, padding bits are used in order to make sure that all characters are transferred as octets. The padding bit should be set to 0. The character is right aligned in the octet with the padding in the bits to the left.

6. Discovery and IrLMP IAS Objects

In normal wired communications, the first step is attaching the cable to the two devices. When using IrCOMM, this is replaced by a three step process of discovery, characterization, and connection, thereby attaching the "infrared cable". The client applications know nothing of this process, so it must be entirely transparent to them. These steps are explained in [IRDAIRLMP], but the IrCOMM unique parts are discussed here. There are two such parts: a new discovery hint bit, and new IAS entries.

6.1 IrCOMM hint bits

IrCOMM makes use of two discovery hint bits:

- print - already defined, indicates that device supports printing services
- IrCOMM - a newly defined hint bit shown in the following table, indicates that the device supports IrCOMM services.

	Byte 2
Bit	Function
8	Telephony
9	File Server
10	IrCOMM
11	reserved
12	reserved
13	reserved
14	reserved
15	Extension

The IrCOMM hint bit must be set for any device that has an IAS object with classname IrCOMM. The printer bit must be set for any device offering the IrLPT IAS object (because IrLPT is confined to printing; explained below). To find out more, a prospective client must query the IAS.

6.2 IrCOMM IAS entry

The IAS is a database of infrared services, a sort of yellow pages listing what a device can provide. An IAS Object consists of a classname and one or more attributes that serve to advertise a service or group of related services on a device (see [IRDAIRLMP] for details). Ideally there would be just one classname for COMM emulation services, but for historical reasons there are two: IrDA:IrCOMM and IrLPT.

The primary IrCOMM IAS entry has classname IrDA:IrCOMM, and at least the following attributes: LsapSel, and Parameters. The LsapSel attribute is needed in order to make a connection (see IrLMP specification). The Parameters attribute allows the client application to distinguish among multiple COMM services, since many different applications can use serial and parallel to communicate. A device should not allow multiple services with identical IAS entries, or the client must in effect toss a coin to decide among them. An additional optional attribute called Instance Name is defined here to help with that exact case.

The following sections show the detailed format of the attributes for IrCOMM IAS entries. (Note - the three components of an IAS entry are not the same as the three components of a parameter described under the Parameters attribute)

6.2.1 LsapSel Attribute

LsapSel (Link Service Access Point Selector) is the unique "address" or id of the service within the context of one device, and is needed to connect to that service. Use of this attribute is mandatory.

If the IrDA:IrCOMM IAS entry is for one or more of the cooked service types (3-Wire, 9-Wire, or Centronics), use the following format.

Attribute Name	Value Type	Description
IrDA:TinyTP:LsapSel	Integer (0x01)	The IrLMP LSAP/TTPSAP of the TTP entity that provides access to the service being advertised Legal values are restricted to the range 0x01-0x6F.

On the other hand, if the IrDA:IrCOMM IAS entry is for 3-Wire raw, use this format instead.

Attribute Name	Value Type	Description
IrDA:IrLMP:LsapSel	Integer (0x01)	The IrLMP LSAP of the service being advertised Legal values are restricted to the range 0x01-0x6F.

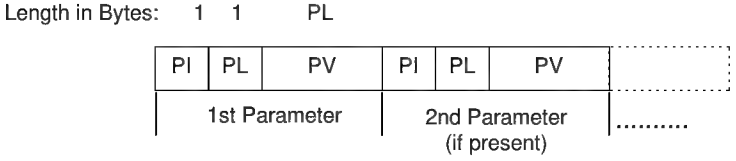
Both LsapSel attributes may be present (e.g. a service can employ both 3-Wire raw and 3-Wire connections), but they must have different values.

6.2.2 Parameters Attribute

The Parameters attribute contains one or more values (themselves called parameters) which characterize the IrCOMM service being provided. They are intended to provide enough information to uniquely identify the service. Use of this attribute is mandatory. The attribute is defined as follows:

Attribute Name	Value Type	Description
Parameters	Octet seq (0x02)	A collection of one or more parameters characterizing an IrCOMM service.

Each parameter in the Parameters attribute consists of a 3-tuple (like those described for the control channel) with size and format as shown in the following diagram:



The maximum length of the Parameters attribute value is 1024 bytes. The format of the elements of the 3-tuple are as follows:

parameter element	Value Type	Description
PI - Parameter Identifier	UINT8	bit 7 set if parameter is critical, bits 0-6 are unsigned integer identifier value. Critical parameter are defined below.
PL - Parameter Length	UINT8	length in bytes of PV
PV - Parameter Value	UINT8 sequence	exact meaning depends on the parameter identifier.

The Parameters attribute collects into one place many characteristics which together define an IrCOMM service. The same information could have been spread into multiple attributes, but that would require multiple IAS GetValueByClass queries, an implementation inconvenience.

Parameters can be marked as "critical" by setting the high bit of the parameter identifier byte (PI in the table below). A critical parameter is described as follows: "if you don't recognize and understand this parameter, you don't want to connect to this service". You are not prevented from connecting, but the point of such parameters is to distinguish specialized services that will only work properly when connected with peers. A general printing or terminal service would probably not want to set any critical parameters, while an IR equipped security system might well use the Fixed-Port-Name critical parameter to encourage connections only from security system control programs.

The IrDA:IrCOMM IAS parameters are shown in the following table, then discussed immediately below that:

PI	PI name	PL	PV datatype		PV Description	PV Default value, notes
0x00	Service Type	1	byte (bitmask)	bit 0 bit 1 bit 2 bit 3	3-Wire raw 3-Wire 9-Wire Centronics	default = highest order bit set in the IAS service type parameter
0x01	Port Type	1	byte (bitmask)	bit 0 bit 1	serial parallel	default = both set
0x02	Fixed Port Name - PI is 0x02, but shows as 0x82 when high bit set, marking it a critical parm.	varies max = 32	byte sequence		Name of fixed port. Normally human readable text, but not required	none.
0x03 - 0x7F, 0x83 - 0xFF	Reserved for future use					

Additional parameters may be defined in future revisions of this specification. Each parameter in described in detail below.

6.2.2.1 Service Type

This parameter indicates what service types are supported (3-Wire raw, 3-Wire, etc.). This parameter is mandatory. At connection time the client will choose the service type from those offered in the service type parameter, selecting a raw service by connecting to the IrDA:IrLMP:LsapSel, or selecting a cooked service by connecting to the IrDA:TinyTP:LsapSel and (if there is more than one cooked service advertised) sending a service type control parameter as described in the previous chapter.

6.2.2.2 Port Type

This parameter indicates the port type (or types) supported (choices are serial and parallel). It is used to narrow the client's IAS search, and potentially to determine the flavor of 3-Wire raw and 3-Wire connections, which can be used for both port types. This is an optional parameter. A modem would generally be a serial device, while a PC would advertise the port type based on the name of the port opened by the application. A printer that cares nothing about any form of control information could advertise both port types. If the parameter is missing, the default is both serial and parallel supported.

6.2.2.3 Fixed Port Name

The fixed port name parameter gives a distinctive name to the associated service (e.g. "XYZ pizza analyzer"), identifying it to like-minded applications while warning off those who do not recognize it. This is an optional parameter - if the service is generic, such as a terminal program, this parameter would not be used. Fixed name is a critical parameter, meaning that devices that don't understand it should not connect to it. The default is no fixed name.

6.2.3 InstanceName Attribute

InstanceName is used to help distinguish among otherwise identical IAS objects. Use of this attribute is optional. The format is as follows.

Attribute Name	Value Type	Description
IrDA:IrLMP:InstanceName	UserString (0x03)	A displayable string to help distinguish among otherwise identical IAS objects.

6.3 Advertising multiple service types

If a service (e.g. printing) can support multiple IrCOMM **service types** (3-Wire, 9-Wire, etc.), it can generally advertise them all under one IAS entry, and the exact service type is then selected using the control channel (or left to default values). However, a special case arises if the printer wishes to advertise both raw and cooked service types (hopefully done only rarely, or at least medium-rarely). A 3-Wire raw service type may not share an LSAP-SEL value with any cooked service type, because raw does not support the control channel and hence there is no way after connection to specify which service to use. This means one service (printing in this example) must somehow advertise two LSAP-SELs. Following are two possible solutions.

- The most compact and efficient solution is to have a single IAS entry with the appropriate service type bits set in the Parameters attribute, and have **both LsapSel attributes contained in this single IAS entry**. If connection is made through the IrLMP:LsapSel it is 3-Wire raw. If it is made through the TinyTP:LsapSel, service type defaults to highest of the cooked types set in the IAS service type parameter, and can be changed with a control channel service type parameter. If both raw and cooked serviced types are combined as recommended here, both types of LsapSel attribute **must** be present in the IAS object.

- Another approach is to have two IAS entries - one for the raw service, and one for any others. The two entries differ by at least the following: the raw entry specifies its LSAP-SEL with the attribute IrDA:IrLMP:LsapSel, and has only the 3-Wire raw service type set in the Parameters attribute (or if a print service, use the IrLPT classname with no Parameters attribute for the same effect). The other entry has the IrDA:TinyTP:LsapSel attribute, and sets all the other service types in the Parameters attribute.

6.4 IrLPT IAS Entry

IrDA:IrCOMM is the preferred classname for IrCOMM IAS entries. However, the classname IrLPT was built into products before the IrCOMM specification existed, and it is supported for backward compatibility. **The definition is fixed, and not intended to be extended**. The service advertised by this entry is **printing via 3-Wire raw service type**.

This IAS object has classname IrLPT, and has a single attribute defined in the following table.

Attribute Name	Value Type	Description
IrDA:IrLMP:LsapSel Note: One existing implementation mis-capitalizes this attribute LSAPSel. For full backward compatibility, check for this spelling also	Integer (0x01)	The IrLMP LSAP of the IrLPT print service Legal values are restricted to the range 0x01-0x6F.

Any device which has this service must also set the printer bit in the Discovery service hint bits.

7. State definition and transitions

This section contains a state chart based on the IrCOMM service primitives described earlier. Descriptions of the states, events and actions are included.

7.1 Start Chart

The state chart for IrCOMM is given below.

Current State	Event	Action(s)	Next State	
IDLE	IrCOMM_Connect.req (lsap, st, qos)	VServiceType = *st* VPeerSAP = lsap *Issue-connect-request*	WAITI	
		Disconnect-Indication	IDLE	
	TTP_Connect.ind (lsap,st)	VPeerSAP = lsap *Connect-Indication*	WAITR	
	LM_Connect.ind(lsap)	VPeerSAP = lsap *Connect-Indication3-Wire*	WAITR	
WAITI	TTP_Connect.cnf	*Connect-Confirmation*	CONN	
	TTP_Disconnect.ind	*Disconnect-Indication*	IDLE	
	LM_Connect.cnf	*Connect-Confirmation*	CONN	
	LM_Disconnect.ind	*Disconnect-Indication*	IDLE	
WAITR	IrCOMM_Connect.rsp	*Issue-Connect-Response*	CONN	
	IrCOMM_Disconnect.req	*Issue-Disconnect-Request*	IDLE	
	TTP_Disconnect.ind	*Disconnect-Indication*	IDLE	
	LM_Disconnect.ind	*Disconnect-Indication*	IDLE	
CONN	IrCOMM_Disconnect.req	*Issue-Disconnect-Request*	IDLE	
	IrCOMM_Data.req	*Issue-Data-Request*	CONN	
	IrCOMM_Control.req	*Issue-Control-Request*	CONN	
	TTP_Disconnect.ind	*Disconnect-Indication*	IDLE	
	TTP_Data.ind	*Process-Data*	CONN	
	LM_Disconnect.ind	*Disconnect-Indication*	IDLE	
	LM_Data.ind	*Data-Indication*	CONN	

7.2 State Definitions

There are four states for IrCOMM protocol machine.

IDLE This is the initial state. No link is established and the IrCOMM user cannot send any data or control parameters.

WAITI The IrCOMM user has requested an IrCOMM connection and is waiting for the confirmation from the other side. Data and control cannot be sent because the connection is not yet established.

WAITR IrCOMM has received an incoming connection request and is waiting for the response from the IrCOMM user. Data and control cannot be sent yet because the connection is not yet established.

CONN IrCOMM is connected to a peer IrCOMM. Data and control can now be sent and received.

7.3 Event Descriptions

IrCOMM_Connect.req(lsap, st, qos) IrCOMM connect request from the user layer. The parameters include the lsap of the remote IrCOMM (lsap), the IrCOMM service type desired (st) and the IrLAP quality of service.

TTP_Connect.ind(lsap,st) Indication of an incoming TinyTP connection from the other side. In the case of IrCOMM this is for 3-Wire, 9-Wire or Centronics service types.

LM_Connect.ind(lsap) Indication of an incoming IrLMP connection from the other side. In the case of the IrCOMM this for a 3-Wire raw connection

TTP_Connect.cnf Confirmation of a TinyTP connection from the other side. This is for cooked type connection (3-Wire, 9-Wire, Centronics).

TTP_Disconnect.ind Indication of a disconnection of a TinyTP connection. This is for cooked type connections.

LM_Connect.cnf Confirmation of a IrLMP connection from the other side. This is for 3-Wire Raw connections.

LM_Disconnect.ind Indication of a disconnection of an IrLMP connection. This is for 3-Wire Raw connections.

IrCOMM_Connect.rsp IrCOMM connect response from the user layer accepting an incoming connection.

IrCOMM_Disconnect.req IrCOMM disconnect request from the user layer either rejecting an incoming connection or requesting to disconnect an existing connection.

IrCOMM_Data.req Request from the user layer to send IrCOMM data

IrCOMM_Control.req Request from the user layer to send IrCOMM control information.

TTP_Data.ind Indication from TinyTP of incoming data from the other side. This is for cooked connections.

LM_Data.ind Indication from IrLMP of incoming data from the other side. This for 3-Wire Raw connections.

7.4 Action Descriptions

VServiceType = *st* Save the service type into the internal variable VServiceType.

VPeerSAP = lsap Save the LSAP into the internal variable VPeerSAP.

Issue-connect-request Issue connect request following the algorithm below.

```
If VServiceType is '3-Wire raw',
        Issue LM_Connect.req(
                called LSAP = VPeerSAP,
                Requested QoS = 'QoS' of IrCOMM_Connect.req,
                Client Data = none
        )

If VServiceType is 'Default',
        Issue TTP_Connect.req(
                called TTPSAP = VPeerSAP,
                Requested QoS = 'QoS' of IrCOMM_Connect.req,
                Calling MaxSduSize = 0,
                Calling UserData = encoded 'InitialControlParameters' of IrCOMM_Connect.req
        )

If VServiceType is not either '3-Wire raw' or 'Default',
        Issue TTP_Connect.req(
                called TTPSAP = VPeerSAP,
                Requested QoS = 'QoS' of IrCOMM_Connect.req,
                Calling MaxSduSize = 0,
                Calling UserData = encoded VServiceType and 'InitialControlParameters' of
                        IrCOMM_Connect.req
        )
```

Disconnect-Indication Give an IrCOMM_Disconnect.ind to IrCOMM user. If 'Reason' of TTP_Disconnect.ind or LM_Disconnect.ind is 'User Request', then 'Reason' of IrCOMM_Disconnect.ind should be 'User Disconnect', else 'Reason' of IrCOMM_Disconnect.ind should be 'Provider Disconnect'. But this rule is not mandatory. Reason may always be set to 'UnSpecified Reason'.

Connect-Indication Give an indication to the IrCOMM user following the algorithm below.

```
If 'ServiceType' exist in the UserData of TTP_Connect.ind,
        VServiceType = 'ServiceType' in TTP_Connect.ind
else
        VServiceType = 'Default'

Notify IrCOMM_Connect.ind(
        CallingLsap = VPeerSAP,
        ServiceType = VServiceType
        InitialControlParameters = Any additional control parameters besides ServiceType in
                TTP_Connect.ind
        QoS = QoS of TTP_Connect.ind
)
```

Connect-Indication3-Wire Give an indication to the IrCOMM user following the algorithm below.

```
Notify IrCOMM_Connect.ind(
        CallingLsap = VPeerSAP,
        ServiceType = 3-Wire-Raw
        QoS = QoS of LM_Connect.ind
)
```

VServiceType = 3-Wire-Raw Set the internal variable VServiceType to 3-Wire-Raw.

Connect-Confirmation Give a connection confirmation to the IrCOMM user (IrCOMM_Connect.cnf).

Issue-Connect-Response Issue a connect response following the algorithm below.

```
If VServiceType is '3-Wire raw' then
        Issue LM_Connect.rsp(
                Calling LSAP = VPeerSAP,
                Confirmation = 'accept'
                Client Data = null
        )
else
        Issue TTP_Connect.rsp(
                Calling TTPSAP = VPeerSAP,
                Called MaxSduSize = 0,
                Called UserData = encoded 'InitialControlParameters' of IrCOMM_Connect.rsp
        )
```

Issue-Disconnect-Request Issue a disconnect request following algorithm given below.

```
If VServiceType is '3-Wire raw' then
        Issue LM_Connect.rsp(
                Calling LSAP = VPeerSAP,
                Confirmation = 'reject'
                Client Data = null
        )
        or Issue LM_Disconnect.req(
                Reason = 'User Request'
                Client Data = 'UserData' of IrCOMM_Disconnect.req
        )
else
        Issue TTP_Disconnect.req(
                UserData = 'UserData' of IrCOMM_Disconnect.req
        )
```

Issue-Data-Request Issue a data request following the algorithm below.

```
If VServiceType is '3-Wire-raw' then
        Issue LM_Data.req(
                UserData = encoded 'Data' of IrCOMM_Data.req
)
else
        Issue TTP_Data.req(
                UserData = encoded 'Data' of IrCOMM_Data.req
)
```

Note: In the case of TTP_Data succeeding 'Data' and 'ControlParameters' may be combined into one 'UserData', also single 'Data' and 'Control Parameters' may be split into multiple (succeeding) 'UserData'

Issue-Control-Request Issue a control request following the algorithm below.

```
Issue TTP_Data.req(
        UserData = encoded 'ControlParameters' of IrCOMM_Control.req
)
```

Note: Succeeding 'Data' and 'ControlParameters' may be combined into one 'UserData', also single 'Data' and 'Control Parameters' may be split into multiple (succeeding) 'UserData'

Process-Data Process the data of a TTP_Data.ind as follows:

```
If control information exists
        Notify IrCOMM_Control.ind(
                ControlParameters = decoded part of control parameters
                        of 'UserData' of TTP_Data.ind
        )

If data exists
        Notify IrCOMM_Data.ind(
                Data = data part of 'UserData' of TTP_Data.ind
        )
```

Data-Indication Give a data indication to the IrCOMM user as follows:

```
Notify IrCOMM_Data.ind(
        Data = 'UserData' of LM_Data.ind
)
```

8. 3-Wire Raw and IrLPT in Detail

The preceding sections introduce all of the concepts needed in IrCOMM. This section and the remaining "<Service type> in Detail" sections bring together all the pieces in one place.

3-Wire raw and IrLPT may be used to emulate either serial or parallel ports in cases where a single exclusive connection is satisfactory. They can emulate **both** port types because there is no control channel, and therefore no information about the non-data circuits of either type is carried - only the data normally flowing through the data circuits is emulated. If data transfer is all a port needs to function, then 3-Wire raw or IrLPT may be fine.

8.1 How 3-Wire raw and IrLPT differ

3-Wire raw and IrLPT are two names for the same COMM emulation service. IrLPT was built into some commercially available devices before this IrCOMM specification was complete, and is included here for compatibility. The intent is that IrCOMM services developed in the future will be advertised under the IrDA:IrCOMM classname rather than IrLPT. Therefore, there are two differences between 3-Wire raw and IrLPT:

- IrLPT has a fixed definition and purpose - it is for printing only. IrCOMM 3-Wire raw can be used for both printing and non-printing tasks
- 3-Wire raw uses an IAS entry with classname IrDA:IrCOMM and at least two parameters. Its IAS definition may be modified or extended over time. IrLPT has classname IrLPT, only one parameter, and the IAS definition is fixed. IAS entry formats are defined in the next section.

Beyond these two distinctions, 3-Wire raw and IrLPT are the same.

8.2 IAS entry and hint bits

An entity advertising 3-Wire raw must set up the IAS entry in one of two forms:

- Classname **IrDA:IrCOMM** with two attributes, called **IrDA:IrLMP:LsapSel**, and **Parameters**. The Parameters attribute has at least the service type parameter with at least the 3-Wire raw bit set in it. The Port type parameter is recommended, and the Fixed port parameter should be used if the service being advertised is very restrictive about connections (should only connect to clients that recognize the fixed port name). Optionally **IrDA:IrLMP:InstanceName** can be used to distinguish between two instances of this service. The Discovery frame must have the IrCOMM hint bit set. It should also have the printer bit set if this is a printing service.
- An alternate entry provided for backwards compatibility with some existing devices has Classname **IrLPT** with just one attribute, called **IrDA:IrLMP:LsapSel**. The printer hint bit must be set in the Discovery frame. This IAS object is only used for printing services.

8.3 Basic link operation

3-Wire raw connections must be exclusive - that is, all other non-IAS connections must terminate before the raw connection is made, and all others must wait until the raw connection is broken before they can connect. This is because 3-Wire raw uses IrLAP flow control, which flow controls off the entire physical link - multiple connections under this scenario could result in deadlock.

At connection, the 3-Wire raw service type can be distinguished reliably by its LSAP-SEL alone, since unlike the cooked types (where multiple service types can be referenced by IrDA:**TinyTP**:LsapSel) only 3-Wire raw can use the LSAP-SEL specified by IrDA:**IrLMP**:LsapSel.

Once connected there is no control channel and no control parameters of any kind to look for - the only data that comes over 3-Wire raw is the user data that would flow over TD and RD on a serial port or the 8 data

lines of a Centronics port. This means 3-Wire raw cannot be used to emulate any kind of hardware handshaking or error reporting.

8.4 Handling the non-data circuits

Even though 3-Wire raw does not transmit the state of the non-data circuits, it must emulate them locally for the application using IrCOMM. In short, the Port Emulation Entity must fake out the application. A reasonable starting point for a DTE is to report DSR, CTS, CD, and RI high during connection. For a DCE, DTR and RTS should be reported high.

A similar approach is applied for IrLPT port emulation.

9. 3-Wire in Detail

3-Wire may be used to emulate either serial or parallel ports in cases where more than one connection may be desirable. 3-Wire can emulate serial or parallel because no information about the non-data circuits of either type is carried - just the bytes normally flowing on the data circuits, and in the serial case some setup and status information. If this is all a port needs to function, then 3-Wire may be fine.

9.1 IAS entry and hint bits

An entity advertising 3-Wire capability has an IAS entry with the following characteristics:

- Classname **IrDA:IrCOMM.**
- **IrDA:TinyTP:LsapSel** attribute indicating the LSAP selector at which the service is located.
- **Parameters** attribute - The Parameters attribute must have the service type parameter with at least the 3-Wire bit set in it. The Port type parameter is recommended, and the Fixed port parameter should be used if the service being advertised is very restrictive about connections (should only connect to clients that recognize the fixed port name).
- **IrDA:IrLMP:InstanceName** attribute is an optional parameter that can be used to distinguish between multiple instances of 3-Wire service.

The Discovery frame must have the IrCOMM hint bit set. It should also have the printer bit set if this is a printing service.

9.2 Basic link operation

Unlike 3-Wire raw, 3-Wire connections can coexist with other non-exclusive IrLMP connections. This is because 3-Wire uses Tiny TP flow control, a method which does not flow control off the entire physical link.

At connection, the 3-Wire service type may not be distinguished reliably by its LsapSel alone; unlike 3-Wire raw it may share use of the IAS LsapSel entry with the other cooked types. The solution is to send the service type control parameter through the control channel, discussed in the next section. This illustrates a fundamental distinction between 3-Wire raw and 3-Wire; in the 3-Wire case, IrCOMM must monitor incoming packets for control channel information.

Despite the presence of the control channel, 3-Wire cannot emulate hardware handshaking, since the non-data circuits are not emulated. If hardware handshaking is required, the 9-Wire or Centronics service types are necessary.

9.3 Control channel usage in 3-Wire

In 3-Wire, the control channel is used for three purposes:

- selecting the service type,
- to exchange **port communication settings** (data rate, data format, and flow control information) when emulating a serial port.
- for certain Type 2 devices to deliver **port line status** (overrun, parity and framing errors) back to Type 1 devices.

The table below defines the parameters used by 3-Wire emulation to deliver the port communication settings and port line status:

PI	PI name	PL	PV datatype		PV Description	PV Default value, notes
0x10	Data rate	4	UINT32, Big-Endian		data rate in Bits/second	undefined
0x11	Data Format	1	byte	bits 0 - 1	Character Length 00 = 5 bits 01 = 6 bits 10 = 7 bits 11 = 8 bits	8 bits, 1 stop bit, no parity
				bit 2	Stop Bits 0 = 1 stop bit 1 = 2 if char len 6,7,8 1.5 if char len 5	
				bit 3	Parity Enable 0 = no parity 1 = parity enabled	
				bits 4 - 5	Parity Type (if enabled) 00 = odd 01 = even 10 = mark 11 = space	
0x12	Flow control	1	byte (bitmask)	bit 0	XON/XOFF on input	none
				bit 1	XON/XOFF on output	
				bit 2	RTS/CTS on input	
				bit 3	RTS/CTS on output	
				bit 4	DSR/DTR on input	
				bit 5	DSR/DTR on output	
				bit 6	ENQ/ACK on input	
				bit 7	ENQ/ACK on output	
0x13	XON/XOFF Flow control characters	2	byte sequence - XON character is first, followed by XOFF character		characters used to represent XON/XOFF	XON - 0x11 XOFF - 0x13
0x14	ENQ/ACK Flow control characters	2	byte sequence - ENQ character is first, followed by ACK character		characters used to represent ENQ/ACK	ENQ - 0x05 ACK - 0x06
0x15	Line Status	1	byte	bit 0	reserved (set to 0)	If the bit is set to 1 the error condition has occurred.
				bit 1	Overrun Error	
				bit 2	Parity Error	
				bit 3	Framing Error	
				bit 4 - 7	reserved (set to 0)	
0x16	Break	1	bitmask	bit 0	Break 0 = Clear break 1 = Set break	sender signals break state
0x17-0x1F	Reserved for future 3-Wire ids					

Each parameter is described below in detail.

9.3.1 Data Rate

The Data rate parameter represents the value set by an application on a device. If this parameter is not sent the data rate is undefined.

9.3.2 Data Format

The Data Format parameter represents the value set by an application on a device. If the parameter is not sent, the default of 8 bit, no parity, 1 stop bit is assumed.

9.3.3 Flow Control

The Flow control parameter represents the flow control type (or types) set by an application on a device. If not sent, the default is no flow control. Some operating systems allow flow control to be set independently for incoming data and outgoing data.

In addition, some operating systems allow multiple flow control mechanisms to be set simultaneously. The Flow Control parameter supports this capability as well.

9.3.4 XON/XOFF and ENQ/ACK Flow Control Characters

These parameters allow different characters to represent XON/XOFF and ENQ/ACK when those types of flow control are used. If not sent, the defaults are 0x11/0x13 for XON/XOFF, and 0x05/0x06 for ENQ/ACK.

9.3.5 Line Status

This parameter allows a Type 2 device to report back error conditions resulting from the line settings issued to it by a Type 1 device. This is particularly important for Type 2 devices are programming an UART based on the control parameters sent from the other side. The errors are:

Overrun Error Received character overwrote and unread character

Parity Error Received character's parity was incorrect.

Framing Error A character (frame) did not terminate with a STOP bit.

9.3.6 Break

This parameter is used to turn the break condition on and off.

9.4 Parameters Sent at Connection Time

At connection time, the initiating IrCOMM must explicitly choose 3-Wire if the highest bit set in the Service Type parameter of the IAS Parameter attribute is **not** 3-Wire (because it defaults to the higher service). The initiating IrCOMM chooses 3-Wire emulation by sending a Service Type parameter in the control channel with only the 3-Wire bit set to 1. This parameter must be sent before any data is sent and can either be sent in the connect request frame or as part of the first data frame. To avoid ambiguity should new service types be introduced in the future, it is a good practice to always send the Service Type parameter if more than one cooked service is available

Up until this point, there is no difference between serial and parallel emulation - both require the service type to be set if there is ambiguity. After the service type is selected (either by default or by receiving a service type control parameter), parallel connections make no more use of the control channel. If they receive any more control information, they can ignore it.

In the serial emulation case, devices must assure that the port communication settings are correctly set. Sending the port communication settings requires sending one or more of the parameters defined above (Data Rate, Data Format, Flow Control, and Flow Control Characters) if the defaults are not acceptable. These settings must be in place before any data is sent. Thus, if the defaults are not acceptable then these parameters should be sent in either the connect/connect-response frame or in the first data frames.

9.5 Parameters Sent During a Connection

During a connection a device must send Data Rate, Data Format, Flow Control, and Flow Control Character parameters whenever these settings change.

During a connection Type 2 devices (especially those that have a UART that has been setup with the port communication settings from another device) are highly recommended but not required to send the Line Status parameter whenever a line status error occurs. Type 1 devices are not required to send this parameter.

9.6 Handling the non-data circuits

Even though 3-Wire does not transmit the state of the non-data circuits, it must emulate them locally for the application using IrCOMM. In short, the Port Emulation Entity must fake out the application. A reasonable starting point for a DTE is to report DSR, CTS, CD, and RI high during connection. For a DCE, DTR and RTS should be reported high.

A similar approach is applied for parallel port emulation.

10. 9-Wire in Detail

9-Wire emulation goes beyond 3-Wire serial emulation by providing infrared pass through of the non-data RS-232 circuits normally used for modem control. Along with the ability to send the status of the non-data circuits comes the need to provide null-modem emulation. Unlike 3-Wire, 9-Wire is only used to emulate serial ports.

10.1 IAS and Hint Bits

An entity advertising 9-Wire capability must have the IAS object with the following characteristics.

- Classname **IrDA:IrCOMM**.
- **IrDA:TinyTP:LsapSel** attribute indicating the LSAP selector at which the service is located.
- **Parameters** attribute containing the Service Type parameter with the 9-Wire bit set. Since 9-Wire is a super set of 3-Wire it is recommended that the 3-Wire bit be set also (this will allow 3-Wire-only devices to connect). It is also recommended that the Port Type parameter be present in the Parameters attribute with the serial bit set to 1 and the parallel bit set to 0 since 9-Wire is only used to emulate serial ports. However, it is legal to have the Centronics and 9-Wire bits both set in the Service Type parameter. In this case, the Port Type parameter (if present) must have both the serial and the parallel bits set to 1 indicating that the service supports both port types.
- **IrDA:IrLMP:InstanceName** attribute is an optional parameter that can be used to distinguish between multiple instances of 9-Wire service.

The IrCOMM hint bit must be set in the hints field of the Discovery frame.

10.2 Basic link operation

Like 3-Wire, 9-Wire uses Tiny TP flow control. Therefore, 9-Wire connections can coexist with IrLMP connections (unlike 3-Wire raw, which must own the link completely)

At connection, the 9-Wire service type may not be distinguished reliably by its LsapSel alone, since it may share the IAS LsapSel entry with other cooked types. The solution is to send the service type control parameter through the control channel, as discussed in the next section. Therefore 9-Wire must monitor incoming packets for control channel information. This is also true of 3-Wire, but 9-Wire has even more control channel traffic to watch for, in particular the states of the non-data circuits.

10.3 Control Parameters

9-Wire employs the control channel to convey information beyond the basic data being sent. 9-Wire supports all the parameters described in 3-Wire serial emulation plus additional parameters required for emulation of the non-data circuits.

The following table describes all the additional control channel parameters used by 9-Wire:

PI	PI name	PL	PV datatype		PV Description	PV Default value, notes
0x20	DTE Line Settings and Changes	1	bitmask	bit 0 bit 1 bit 2 bit 3	Delta DTR Delta RTS DTR State RTS State	Delta 0 = circuit not changed 1 = circuit changed State 0 = state is low 1 = state is high
0x21	DCE Line Settings and Changes	1	bitmask	bit 0 bit 1 bit 2 bit 3 bit 4 bit 5 bit 6 bit 7	Delta CTS Delta DSR Delta RI Delta CD CTS State DSR State RI State CD State	Delta 0 = circuit not changed 1 = circuit changed State 0 = state is low 1 = state is high
0x22	Poll for Line Settings	0	no data			sender requests line settings and changes. Can be sent by either DTE or DCE.
0x23 - 0x2F	reserved					

Each parameter is described in detail below.

10.3.1 DTE Line Settings and Changes

This parameter is used by a DTE to tell the other side the initial values of DTR and RTS and to indicate changes in the status of these circuits. The State bits indicate the current state (after the delta has been applied) of DTR and RTS. When used immediately after connection to indicate the initial state, the delta bits are set to 0. When used later to indicate that a circuit has changed state, the delta bit is set to 1 for the changed circuit. This parameter is also sent as a response to a poll. In this case the delta bits should be set to 0.

10.3.2 DCE Line Settings and Changes

This parameter is used by a DCE to tell the other side the initial values of DSR, CTS, CD and RI and to indicate changes in the status of these circuits. The State bits indicate the current state (after the delta has been applied) of DSR, CTS, CD and RI. When used immediately after connection to indicate the initial state, the delta bits are set to 0. When used later to indicate that a circuit has changed state, the delta bit is set to 1 for the changed circuit. This parameter is also sent as a response to a poll. In this case the delta bits should be set to 0.

10.3.3 Poll for Line Settings

This parameter is sent by an entity to request the current state of the line settings controlled by the other side. A DTE should respond with a DTE Line Settings parameter and a DCE should respond with a DCE Line Settings parameter.

10.4 Parameters Sent at Connection Time

The first control channel use is at connection time (just as in the 3-Wire case), when the initiating IrCOMM may need to indicate which service it is intending to use. This is required if the highest bit set in the Service Type parameter of the IAS Parameter attribute is not 9-Wire. The initiating IrCOMM must then choose 9-

Wire emulation by sending a Service Type parameter in the control channel with only the 9-Wire bit set to 1. This parameter must be sent before any data is sent and can either be sent in the connect request frame or as part of the first data frame. To avoid ambiguity should new service types be added in the future, this parameter should be sent whenever connection is made to a service offering more than one cooked service type.

9-Wire devices must properly communicate the port communication settings as described in the 3-Wire in detail chapter (Data Rate, Data Format, Flow Control and Flow Control Characters). In addition they must send the initial state of the non-data circuits using either **DTE line settings** or **DCE line settings** (defined in the table above). The delta bits should be set to 0. These parameters must be sent either in the connect/connect- response frame or in the first data frames

After service type, the port communication settings, and initial line settings, the 9-Wire connection can finally send data.

10.5 Parameters Sent During a Connection

During a connection parameters are sent as follows:

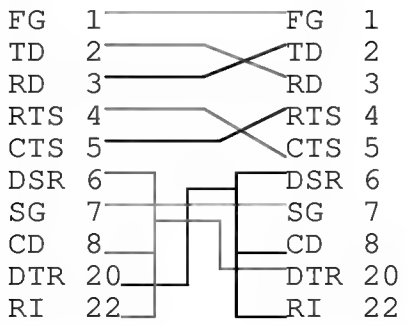
1. Devices must send line-settings-and-changes parameters whenever a non-data circuit changes state, and whenever it receives a Poll for Line Settings from the other side. A DTE will send the DTE Line Settings and Changes parameter while a DCE will send the DCE Line Settings and Changes parameter.
2. Devices must send Data Rate, Data Format, Flow Control, and Flow Control Character parameters whenever these settings change.
3. Devices must send the appropriate Break parameter when the break condition occurs
4. Type 2 devices (especially those that have a UART that has been setup with the port communication settings from another device) are highly recommended but not required to send the Line Status parameter whenever a line status error occurs. Type 1 devices are not required to send this parameter.

10.6 Null Modem Emulation

When two DTEs are to be connected directly by wire without modems between them, some conflicts arise:

- Connectors may not match.
- Circuit mismatch - each side receives data on the circuit it normally uses to send, and gets nothing on the circuit it normally receives data on. For instance, one side sends bits out TD, and the receiving side gets them on TD (instead of RD, where it would normally get them if modems were involved).

These problems are normally overcome with a null modem cable, or at least with a series of experiments involving a breakout box, multiple cables, connectors, and paper clips. The primary function of the null modem is to switch the transmitted and received data lines, so that bits sent out TD on one side appear at RD on the other. Then the other non-data circuits can be hooked together in a variety of ways to convince the communication programs that nothing is amiss. This can range from completely faking it locally (connect local RTS to CTS and DTR to DSR, CD, and RI), to any number of additional switches of wires in the cable (e.g. local RTS to remote CTS). The latter cases are needed for various forms of hardware flow control to work. An example of one type of null modem is shown below.



IrCOMM automatically solves most of the DTE to DTE problems:

- IR removes the cable, so connectors are not a problem
- The frame format allows IrCOMM to assume that all data is coming in on RD. This takes care of the switching of RD and TD that a null modem cable provides.

What remains are the non-data circuits. If an IrCOMM DTE sees DTE line-change-settings-and-changes coming in, it knows that a DTE to DTE connection exists, and it uses a **Local Null Modem Emulator** to solve this last problem.

Each IrCOMM DTE must have a Local Null Modem Emulator which translates incoming DTE line-settings-and-changes to DCE line-settings-and-changes. For this Null Modem Emulator to work IrCOMM DTEs must follow the rules given below.

- IrCOMM DTE must send initial line settings using the DTE Line Settings and Changes control parameter (which consists of the states of RTS and DTR).
- When an IrCOMM DTE receives a DCE Line Settings and Changes parameter it sets the local state of CTS, DSR, CD and RI according to the values found in this parameter.
- When an IrCOMM DTE receives a DTE Line Settings and Changes parameter it feeds this parameter to the Local Null Modem Emulator which will translate the states of RTS and DTR (that came in the parameter) to appropriate states for CTS, DSR, CD and RI.

Unfortunately, no single null-modem cable wiring scheme works in all cases. As a result, it is not possible to give definitive rules governing the translation performed by the Local Null Modem Emulator. As a start, several sources suggest the following scheme:

- Set local CTS according to incoming RTS.
- Set local DSR CD, and RI according to incoming DTR.

The fact of the matter is that this may not work; the important thing is to find out what the local communication program needs. Implementations of IrCOMM may want to provide a means for users to configure the Local Null Modem Emulator.

11. Centronics in Detail

IrCOMM Centronics emulation provides a means for emulating a non-IEEE 1284 parallel interface or an IEEE 1284 parallel interface that uses Nibble, Byte, ECP, or EPP mode(s) for bi-directional communications.

11.1 IAS and Hint Bits

An entity advertising Centronics capability must have the IAS objects with the following characteristics.

- Classname **IrDA:IrCOMM**.
- **IrDA:TinyTP:LsapSel** attribute indicating the LSAP selector at which the service is located.
- **Parameters** attribute containing the Service Type parameter with the Centronics bit set. It is recommended that the Port Type parameter be present in the Parameters attribute with the parallel bit set to 1 and the serial bit set to 0 since the Centronics service type is only used to emulate parallel ports. However, it is legal to have the Centronics and the 9-Wire bits both set in the Service Type parameter. In this case, the Port Type parameter (if present) must have both the parallel and serial bits set to 1 indicating that the service supports both port types.
- **IrDA:IrLMP:InstanceName** attribute is an optional parameter that can be used to distinguish between multiple instances of Centronics service.

The IrCOMM hint bit must be set in the hints field of the Discovery frame.

11.2 Basic link operation

Centronics emulation connections can coexist with other non-exclusive IrLMP connections because TinyTP is used as the flow control mechanism.

Centronics IrCOMM emulates either a non-IEEE 1284 parallel interface or an IEEE 1284 parallel interface using the Nibble, Byte, ECP, or EPP modes.

11.2.1 Traditional or compatible parallel interface emulation

For the non-IEEE 1284 emulation devices, the parallel emulation will act like the traditional uni-directional parallel interface. The only status information that can be retrieved is the traditional unidirectional parallel status information such as Paper Error and the general Error status.

11.2.2 IEEE 1284 emulation

A full IEEE 1284 implementation is also possible through the Centronics emulation of IrCOMM. The four bi-directional modes are emulated: Nibble, Byte, ECP, and EPP.

For Type 1 devices, Nibble and Byte mode support is indistinguishable in function level support. ECP mode emulation allows data to be directed to different channel numbers and EPP mode emulation allows a full bus extension implementation complete with address and data accessible.

For Type 2 devices, the modes that are supported are the intersection of the capability of the Type 2 device and capability of device(s) that are parallel attached to the Type 2 device.

For a detailed description of the IEEE 1284 interface, refer to [IEEE1284].

11.3 Centronics control channel parameters

The control channel provides a means to convey information that is important for emulating a parallel port and that is separate from the user data being transmitted between the devices.

The list of parameters that can be issued to a Type 1 or Type 2 device under IrCOMM is as follows:

PI	PI name	PL	PV datatype	PV Description	PV Default value, notes
0x30	Status query	0	no data		Requests status of parallel lines
0x31	Set Busy Time-out	1	Byte (value)	Time-out value in seconds	Default busy time-out value is 0 seconds (Disabled).
0x32	IEEE 1284 Mode Support	0	no data		Requests IEEE 1284 communication modes supported
0x33	IEEE 1284 Device ID	0	no data		Requests IEEE 1284 Device ID.
0x34	Select IEEE 1284 Mode	varies	First Byte (value): 0x01 0x02 0x04 0x08 0x10 0x20 Second byte is reserved for Extensibility Link	 Compatible Nibble Byte ECP without RLE ECP with RLE EPP	Specifies IEEE 1284 Mode. Default IEEE 1284 mode is Compatible.
0x35	IEEE 1284 ECP/EPP data transfer	2	First byte (value) 0x10 0x11 0x20 0x21 Second byte (value)	 ECP without RLE ECP with RLE EPP Read EPP Write If ECP mode, channel number. If EPP mode, address	

The list of responses that can be returned by a Type 1 or Type 2 device under IrCOMM is as follows:

PI	PI name	PL	PV datatype	PV Description	PV Default value, notes
0x38	Status query response	1	Byte (bitmask) bit 0 bit 1 bit 2 bit 3	 = 1 if the Time-out for peripheral busy has expired. = 1 if IO Error (/FAULT) is active = 1 if Selected (SELECT) line is active = 1 if Paper End (Paper Empty) line is active	Response to Status Query or returned whenever Status has changed.
0x39	Set Busy Time-out response	1	Byte (value) 0x00 0x01	 Time-out value accepted Time-out not supported	
0x3A	IEEE 1284 Mode Support	1	Byte (bitmask) bit 0 bit 1 bit 2 bit 3 bit 4 bit 5	 Compatible Nibble Byte ECP without RLE ECP with RLE EPP	IEEE 1284 communication modes supported.
0x3B	IEEE 1284 Device ID	varies	First byte (bitmask) bit 0 Remaining bytes contain IEEE 1284 Device ID	 Set to 1 equals last packet	Response to IEEE 1284 Device ID query.
0x3C	Select IEEE 1284 Mode	1	Byte (value): 0x00 0x01	 Request successful Request denied: Mode not supported	
0x3D	IEEE 1284 ECP/EPP data transfer	2	First byte (value) 0x10 0x11 0x20 0x21 Second byte (value)	 ECP without RLE ECP with RLE EPP Read EPP Write If ECP mode, channel number. If EPP mode, address	

11.3.1 Status Query

The Status Query command can be used by a computer to determine the status of the emulated parallel port. The values returned in the Status Query response from a peripheral are values derived from traditional parallel interface implementations. The Status Query response can either be returned as a solicited response to a Status Query command or as an unsolicited response any time the status value changes [after the infrared connection has been established]. Paper End, Error, and Selected status values are derived from the lines in the traditional parallel connector referred to as Perror, nFault, and Select respectively. The busy time-out condition in the Status Query has traditionally be generated when a peripheral has been busy too long.

11.3.2 Set Busy Time-out

Traditionally the parallel port Busy Time-out has been implemented by the host computer. It has been used to report an error condition to the user that something is wrong with the peripheral because it has been "busy" too long. With IrCOMM, the host computer does not have access to the hardware lines to perform the same time-out function as with the traditional parallel interface. Therefore this function is moved to the device that is emulating the parallel port. It is within the prerogative of the peripheral to not implement the time-out function and notify the computer in the Set Busy Time-out response that the time-out function is not supported.

If the time-out value is set to zero (0) then the time-out is disabled. When the value is non-zero, the time-out bit is set in the Status Query response if the parallel device has been "busy" for the time-out period without any other error conditions being detected.

11.3.3 IEEE 1284 Mode Support

A Type 1 device may report the IEEE 1284 mode that best suits the function for that device. Nibble or Byte modes are functionally equivalent for a Type 1 device. ECP mode provides different control channels or registers that can be used for various purposes. EPP mode provides bus extension capability to the parallel port by incorporating address and data read/write capabilities.

A Type 2 device must only report the modes negotiated between the Type 2 device and the parallel attached device.

11.3.4 IEEE 1284 Device ID

IEEE 1284 provides for the capability of a device returning an IEEE 1284 Device ID. The IEEE 1284 Device ID is returned in the control channel field. If a Device ID is not supported, the peripheral returns the IEEE 1284 Device ID response without a Device ID. The last packet bit is set to a 1 if the Device ID is contained in one and only packet or is set to 1 on the last packet if the Device ID is split across multiple packets.

11.3.5 Select IEEE 1284 Mode

This command allows the computer to select the IEEE 1284 mode that it wants to use from the modes returned in the IEEE 1284 Mode Support response. The proper ECP or EPP mode must be selected by this command before the IEEE 1284 ECP/EPP data transfer command is issued to a peripheral. This command must also be used to select a bi-directional IEEE 1284 mode before a peripheral will return user data responses to the host computer. In other words, Compatible mode is selected if a host computer does not want to or is not sent up to receive any user data responses. This operation is exactly like a regular parallel interface. The host computer must select a bi-directional IEEE 1284 mode before receiving any responses from a peripheral.

11.3.6 IEEE 1284 ECP/EPP data transfer

Before processing any ECP or EPP type data, a peripheral must know the channel number in case of ECP data and must know the address for any EPP data. Any data is still contained in the user data field. In the case of an EPP write parameter, the address that is to be written to would be contained in the control channel and the data that is to be written is contained in the user data. For an EPP read parameter, the data read from the

specified address would be contained in the user data field on the response. On an ECP with RLE parameter, the RLE encoded data would be transmitted in the user data field.

12. Annex A IR Terminal Adapter (IrTA)

Infrared communication channel enables various application programs running on personal computers and PDAs to communicate with other application programs without hard-wired cables. These application programs often require connection to remote systems via non-infrared network such as conventional, asynchronous character-based data stream channel with public switched telephone network (PSTN) or integrated service digital network (ISDN) using DCEs such as modems and terminal adapters. In this case, data stream channel between applications and DCE (Data Circuit Equipment) have to be established and DCE should be controlled over infrared communication channel.

For this purpose, this annex defines an equipment called IrTA (Infrared Terminal Adapter) which interconnect infrared data channel with DCE signal line, and describes its specifications and requirements based on the IrCOMM protocol stack and how IrTA is controlled.

12.1 Model and components

IR-DTE is a data terminal that has infrared communication capability based on IrDA standard(IrDA SIR, IrLAP, IrLMP, Tiny TP, IrCOMM). The IR-DTE communicates to a data terminal on the other side of the line through infrared medium and public network. IrTA is a terminal adapter that relays control sequence and data stream between IR-DTE and DCE. IrTA is connected to IR-DTE through IrCOMM service interface and directly connected to DCE via ITU-T V series interface. IR-GW is a system that connects IR-DTE to the public network. IR-GW is composed of IrTA and DCE.

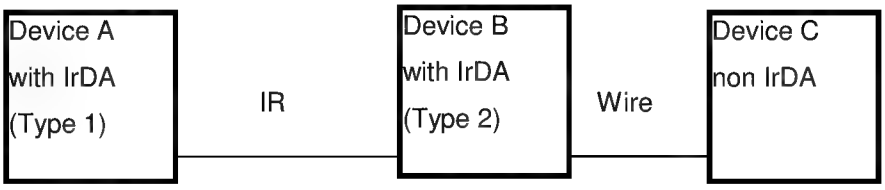
Compared to the definition of device type described earlier,

IR-DTE is a type 1 device

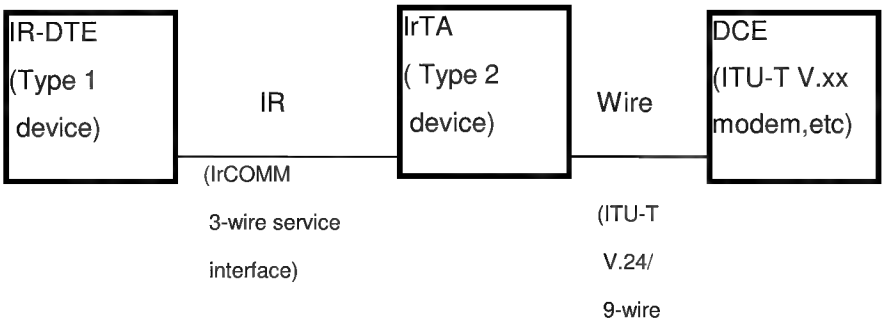
IrTA is a type 2 device

The diagram below illustrates the diagram of IrTA.

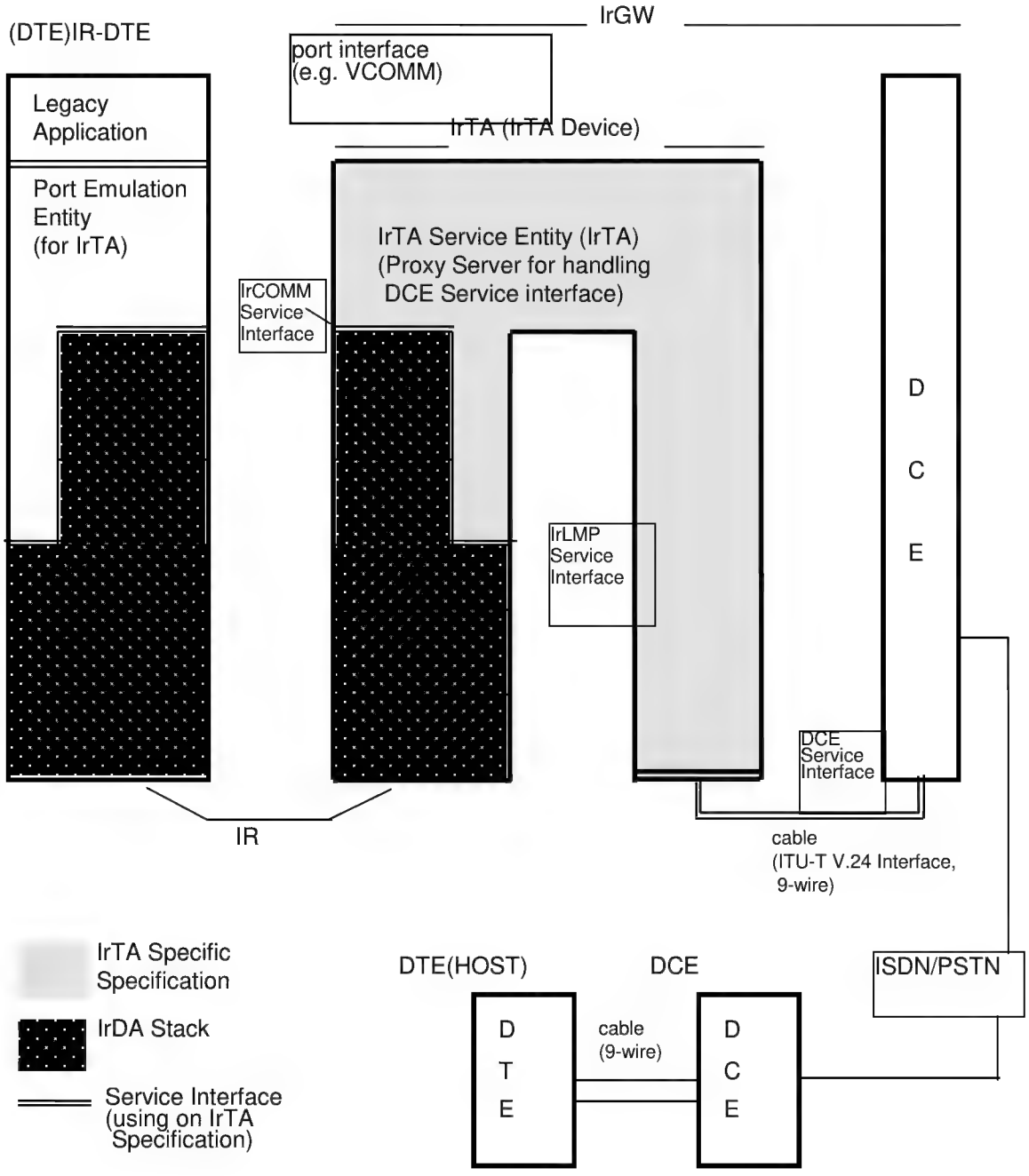
[The diagram of IrCOMM Type 1 and Type 2 device]



[The diagram of IrTA]



The diagram below shows the communication model for the specifications.



12.1.1 IR-DTE

IR-DTE is an end system of infrared data communication and consist of these protocols layers and components. The followings are defined based on the reference model for IrCOMM service definition (see section 3 of this document).

* Legacy Application

See section 3 of this document.

* Port Emulation Entity

See section 3 of this document.

In the case of the IR-DTE connect to IrTA, it is necessary for the Port Emulation Entity to control the IrTA Service Entity, described later, by using both IrCOMM and IrLMP.

* IrCOMM

See section 3 of this document.

* TinyTP, IrLMP, IrLAP, SIR

Infrared communication protocol layers defined by IrDA.

12.1.2 IrTA

From the infrared data communication point of view, IrTA is an end system. From the DTE-DCE data communication point of view, IrTA act as an DTE. IrTA consists of these protocols layers and components.

Note; After clause 12.2, IrTA Service Entity indicates IrTA and the equipment of IrTA show the IrTA device.

* IrTA Service Entity

An entity which performs such as,

Relays data between infrared channels and a V.24 interface linked to DCE

Set communication parameters of V.24 interface linked to DCE directed by IR-DTE

In other words, IrTA Service Entity is an proxy server which handles requests from a Port Emulation Entity in IR-DTE and relays data between a Port Emulation entity in IR-DTE and a DCE.

* IrCOMM

See section 3 of this document.

* TinyTP, IrLMP, IrLAP, SIR

Infrared communication protocol layers defined by IrDA.

* DCE

DCE is a modem for the public network, defined by the ITU-T, V-series Recommendations or a TA (terminal adapter) achieves asynchronous communication over ISDN using DTE supporting the ITU-T, V-series Interface ([ITU-TV24]). In this annex, the following functions should be defined.

* Automatic calling, call incoming and control sequence for

achieving DCE control by character sequence over the mutual

connection between TA and DTE. The precise control sequences

are not the scope of this standard.

The partner terminal on the public network has the same conditions as DCE above. It is usually a modem defined by the ITU-T, V-series Recommendations, or a TA (terminal adapter) that achieves asynchronous communication over ISDN using a DTE supporting the ITU-T, V-series Interface ([ITU-TV24]).

The actions of the PSTN or ISDN are terminated by the commands of the DCE and signal sequences, and therefore should be outside the scope of this standard.

12.1.3 Interface

The definition of Service Interfaces among IR-DTE and IrTA should follow the section 3 of this document. However, in the case of IrTA, the DCE Service Interfaces between IrTA Service Entity and DCE are used. Detail of these interfaces are shown in the following clause.

Furthermore, regarding to the detail description on the Service Interface (Port interface) dedicated to the Legacy Application, provided by the Port Emulation Entity of the IR-DTE is not included in this standard and it is matter of implementations.

12.2

IrTA specific requirements

12.2.1 Requirements for Port Emulation Entity in IR-DTE

* Port Emulation Entity always makes a connection with IrTA Service Entity as an initiator.

* Port Emulation Entity must be able to connect to a 3-wire service type in IrCOMM.

* Port Emulation Entity can receive the signal status such as Break, parity error, framing error, over run error sending by IrTA, and can be reflected to it's processing.

For example; to indicate the error status based on the request from Legacy Application, etc.

* In the Port Emulation Entity, the Flow control characters (XON, XOFF) can be transparent.

* After the link establishment of IrCOMM, the Port Emulation Entity indicate status "ON" of the DSR, CTS, CD, RI, except TD, RD signal line for V.24 interface to the Legacy Applications, and indicate status "OFF" of the DSR, CTS, CD, RI signal line in the case of disconnection of or under making a connection of IrCOMM.

Legacy Application can recognize the IrCOMM link disconnection, by receiving the status of DSR signal line from ON to OFF. When Port Emulation Entity received the status change request of DTR signal from "ON" to "OFF" from Legacy Application, it execute the link disconnection of IrCOMM.

12.2.2 Requirements for IrTA

* IrTA must not makes a connection with Port Emulation Entity on IR-DTE as an initiator.

* IrTA must return "Modem" and "IrCOMM" bits as Service Hints of LM_DiscoverDevices.

* If IrTA is asked for a IAS entry of "IrDA:TinyTP:LsapSel", it must return a LsapSel value with the parameters; ServiceType: 3-wire, PortType: COM.

* When IrTA receives port communication settings on control channel, it must set up itself according to them and send the same parameters back to Port Emulation Entity on IR-DTE in order to confirm the settings.

If IrTA cannot set up itself on requested parameters, it must keep the previous settings and send the previous ones to Port Emulation Entity on IR-DTE.

* Just after the link establishment of IrCOMM (when "IrCOMM_Connect.indication", described in clause 12.3, is received and "IrCOMM_Control.indication" is not received), and user data are received, port communication settings of the communication interface(cable) between IrTA and DCE are applied the following default settings.

Data rate: 9600bps Flow control: RTS/CTS (on input and output)

Other Settings: default Value of control parameters in IrCOMM

* IrTA sends flow control characters (XON, XOFF) received from port emulation entity of IR- DTE transparently. But when XON/XOFF flow control method are chosen in IrTA and flow control characters are received from DCE, IrTA does not send flow control characters to IR-DTE, and does flow control according to flow control characters. On the other hand, when RTS/CTS flow control method without XON/XOFF is chosen, IrTA sends flow control characters to IR-DTE transparently, and does flow control according to status of CTS circuit.

Furthermore, when IrTA can not send data to port emulation entity of IR-DTE cause to flow control status of IrCOMM, IrTA dams up data from DCE by using flow control character or RTS circuit.

Besides in IrTA, ENQ/ACK flow control method and DTR/DSR flow control method are out of scope in this annex.

12.2.3 Requirements for DCE

* DCE must support the following settings.

DTE speed: 300, 600, 1200, 2400, 4800, 9600, 19200 bps

Data Format: 8N1, 7E1, 7O1, 7N2 (character length, parity, stop bit)

* The communication interface(cable) between IrTA and DCE is based on the 9-wire on ITU-T V.24 recommendation.

* 9-wire		106	Clear to Send(CTS)
102	Signal Common(GND)	107	Data Set Ready(DSR)
103	Transmitted Data(TD)	108/2	Data Terminal Ready(DTR)
104	Received Data(RD)	109	Carrier Detect(CD)
105	Request to Send(RTS)	125	Ring Indicator(RI)

* DCE execute the disconnection of the network (PSTN/ISDN) by changing the DTR signal from "ON" to "OFF".

12.3 Service Definition

The following specify the service primitives used among Port Emulation Entity, IrCOMM, IrLMP, IrTA Service Entity and DCE. Here, the following diagram shows the reference model corresponding to the service definition of section 3 of this document.

12.3.1 Service Elements Between Port Emulation Entity and IrLMP in the IR-DTE

Port Emulation Entity on IR-DTE uses the following mandatory service elements served by IrLMP (IrLMP Service Interface) to discovering IrTA device.

* LM_DiscoverDevices
* LM_GetValueByClass

Refer to [IRDAIRLMP] for details of each service element of IrLMP.

12.3.2 Service Elements Between Port Emulation Entity and IrCOMM in the IR-DTE

Port Emulation Entity on IR-DTE uses the following mandatory service elements served by IrCOMM (IrCOMM Service Interface) to control IrTA Service Entity to communicate with DCE.

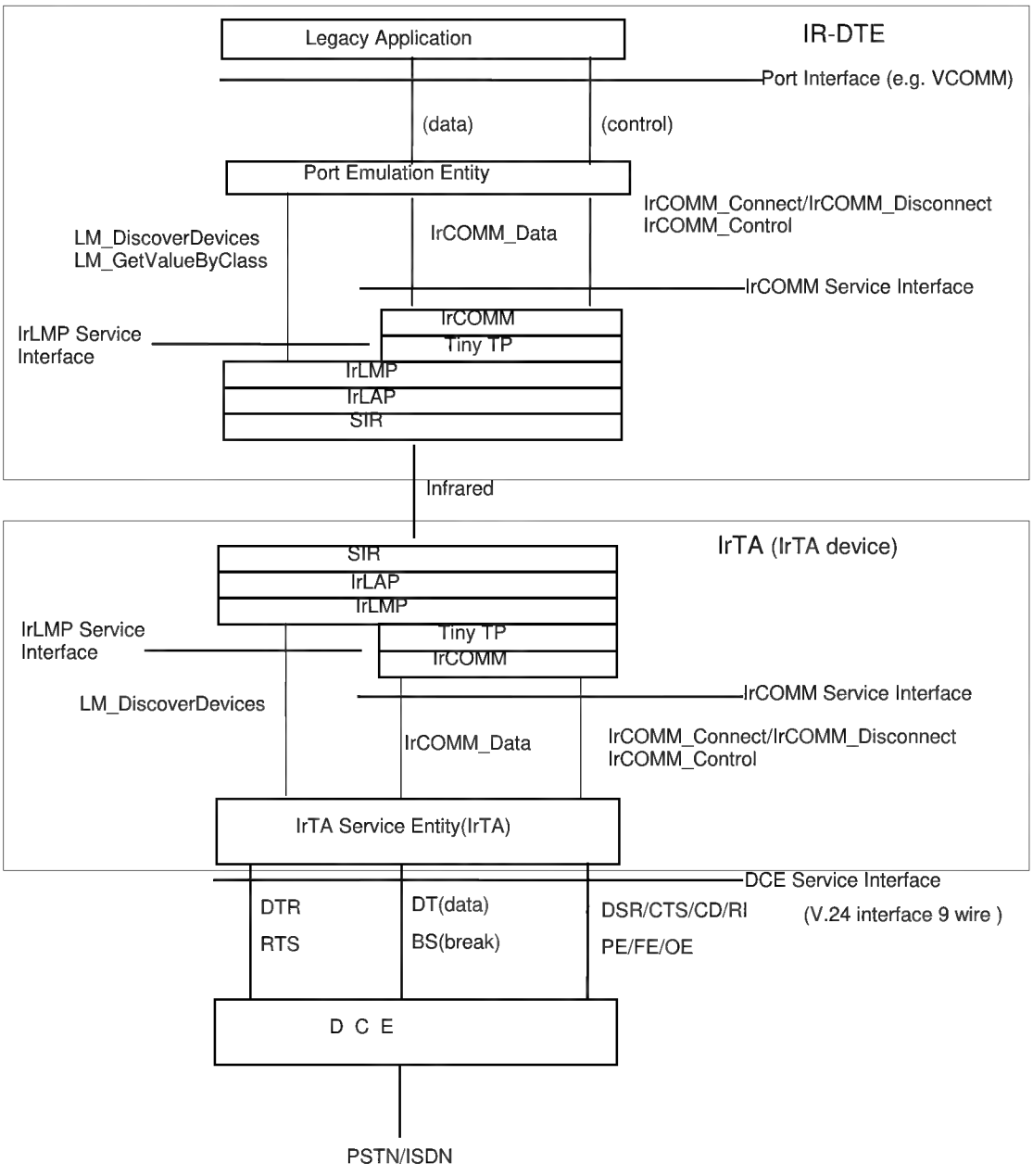
* IrCOMM_Connect
* IrCOMM_Disconnect
* IrCOMM_Data
* IrCOMM_Control

Refer to section 3 for details of each service element of IrCOMM.

12.3.3 Service Elements Between IrTA and IrLMP in the IrTA device

IrTA uses the following mandatory service elements served by IrLMP (IrLMP Service Interface) to be discovered by IR-DTE.

* LM_DiscoverDevices



12.3.4 Service Elements Between IrTA and IrCOMM in the IrTA device

In IrTA Service Entity, the communication for infrared section, (the communication with IR-DTE) uses the mandatory service elements of IrCOMM shown by 12.3.2.

12.3.5 Service Elements Between IrTA and DCE

DCE serves IrTA the following services and these services elements (DCE Service Interface) are used in IrTA.

12.3.5.1 DT (Data)

DT.request (data)

DT.indication (data)

data user data

The DT.request service element is used to transmit the user data to DCE.

The DT.indication service element is used to receive the user data from DCE.

12.3.5.2 DTR (Data Terminal Ready)

DTR.request (status)

status ready/not ready

The DTR service element is used to notify the DCE that IrCOMM connection has been established and released between IrTA and IR-DTE.

Status becomes "ready" after IrCOMM connection is established. And then status becomes "not ready" after IrCOMM connection is released. The mapping is done to DTR in V.24 interface.

DTR.request(ready): Turning ON circuit DTR

DTR.request(not ready): Turning OFF circuit DTR

12.3.5.3 DSR (Dataset Ready)

DSR.indication (status)

status ready/not ready

The DSR service element is used to notify IrTA of the state of the DCE. Status becomes "ready" when the DCE is possible to be used, and becomes "not ready" when the DCE is possible not to be used. The mapping is done to DSR in V.24 interface.

DSR.indication(ready): Turning ON circuit DSR

DSR.indication(not ready): Turning OFF circuit DSR

12.3.5.4 RTS (Request to Send)

RTS.request (status)

status ready/busy

The RTS service element is used to notify the DCE device whether IrTA is able to receive the data from the DCE. Status becomes "ready" when it is possible to receive, and when it is not possible to receive, status becomes "busy". The mapping is done to RTS in V.24 interface.

RTS.request(ready): Turning ON circuit RTS

RTS.request(busy): Turning OFF circuit RTS

12.3.5.5 CTS (Clear to Send)

CTS.indication(status)

status ready/busy

The CTS service element is used to notify IrTA whether the DCE is able to receive the data from IrTA. Status becomes "ready" when it is possible to receive, and when it is not possible to receive, status becomes "busy". The mapping is done to CTS in V.24 interface.

CTS.indication(ready): Turning ON circuit CTS

CTS.indication(busy): Turning OFF circuit CTS

12.3.5.6 CD (Carrier Detect)

CD.indication(status)

status on/off

The CD service element is used to notify the IrTA whether the DCE may detect the carrier. Status is turning on when the carrier is detected, and when the carrier is dropped, status is turned off. The mapping is done to CD in V.24 interface.

CD.indication(on): Turning ON circuit CD

CD.indication(off): Turning OFF circuit CD

12.3.5.7 RI (Ring Indicate)

RI.indication(status)

status on/off

The RI service element is used to notify the IrTA whether the DCE may detect the Incoming call . Status is turning on when the incoming call is detected, and when the carrier is dropped, status is turned off. The mapping is done to RI in V.24 interface.

RI.indication(on): Turning ON circuit RI

RI.indication(off): Turning OFF circuit RI

12.3.5.8 BS (Break Signal)

BS.request (status)

BS.indication (status)

status set/clear

The BS.request service element is used to transmit status of the Break Signal start(set) or the Break Signal end(clear) to DCE.

The BS.indication service element is used to receive status of the Break Signal start(set) or the Break Signal end(clear) from DCE.

12.3.5.9 OE (Overrun Error)

OE.indication

The OE service is a internal event used to notify the IrTA whether Overrun Error occurs in V.24 interface.

12.3.5.10 PE (Parity Error)

PE.indication

The PE service is a internal event used to notify the IrTA whether Parity Error occurs in V.24 interface.

12.3.5.11 FE (Framing Error)

FE.indication

The FE service is a internal event used to notify the IrTA whether Framing Error occurs in V.24 interface.

12.3.5.12 changeDTEstatus (Change DTE Status)

changeDTEstatus(status)

status ready/busy

This status is a internal event used to notify the change (enable or unable) of transmitting status from IrTA to IR-DTE. Status is ready when transmitting status becomes enable and status becomes BUSY when unable.

12.4 State Transition Description of IrTA

The description of the state transition in this annex and the example of sequence of section 12.5 are the specification of the IrCOMM link disconnection based on the DCE disconnection from Network.

12.4.1 General Description

The IrTA is responsible for the mapping between the primitives of IR-DTE and DCE. IrTA controls its flow control functions between IR-DTE and IrTA and between IrTA and DCE independently of each other.

12.4.2 Status Machine Rules

In the section below, precise description of the procedures are specified using state transition diagrams, state transition table and textual descriptions. Any ambiguities of textual descriptions should be accurately defined by state transition diagrams or state transition tables.

In the each column of state transition table, the state transition rule is described using the following notation. (where the P means the predicate, A and A' means action, and S and S' are the states.)

A => S or

if P **then**
A => S
else
A' => S'

The first notation means that state moves to S unconditionally after action A is executed. The second one means that if the predicate P is true then state moves to S after action A is executed, and if the predicate P is false then state moves to S' after action A' is executed.

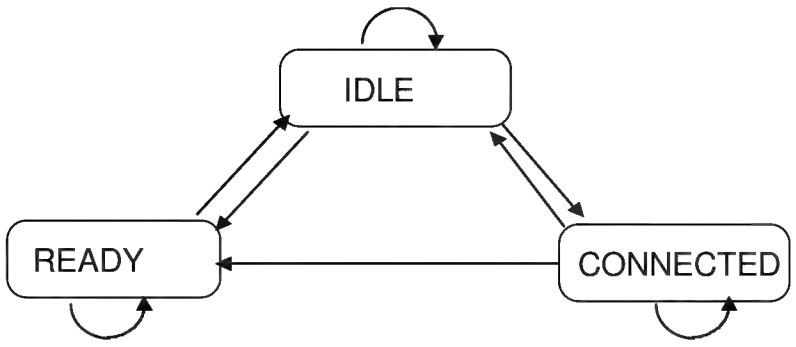
And the following logical operators and a set operator are commonly used:

U Set Union

^ Logical ANDing of predicates

<- Set membership predicate: a <-A is true if a is an element of set A.

12.4.3 IrTA State Transition Table



12.4.3.1 State Transition Table (Common)

EVENT	STATE		
	IDLE	READY	CONNECTED
LM_DiscoverDevices. indication(addr,info,method)	{Addr, Info, Method} = {Addr, Info, Method} U {addr, info, method} =>IDLE	{Addr, Info, Method} = {Addr, Info, Method} U {addr, info, method} => READY	{Addr, Info, Method} = {Addr, Info, Method} U {addr, info, method} => CONNECTED
IrCOMM_Connect. indication (CallingLSAP=sap-id, ServiceType=type, QoS=qos)	IrCOMM_Disconnect. request (Reason='User Disconnect', UserData=data) => IDLE	if Connected = ø ^ type <-{ServiceType} then IrCOMM_Connect. response() DTR.request(ready) => CONNECTED else IrCOMM_Disconnect. request(Reason= 'User Disconnect', UserData=data) =>READY	IrCOMM_Disconnect.request (Reason='User Disconnect', Userdata=data) => CONNECTED
IrCOMM_Disconnect. indication (Reason=reason, UserData=data)	Error => IDLE	Error =>READY	DTR.request(not ready) => READY
IrCOMM_Control.indication (ServiceType=type)	Error => IDLE	Error => READY	if General_Control_parameters = accepable ^ type <-{DefaultServiceType} then SetServiceType(Curtype=type) => CONNECTED else Error => CONNECTED

EVENT	STATE		
	IDLE	READY	CONNECTED
IrCOMM_Control.indication (Datarate=rate, Dataformat=format, Flowcontrol=flow, XON/XOFF=xchars, ENQ/ACK=echars)	Error => IDLE	Error => READY	if rate<-{enable Data rate} then SetDataRate(Currate=rate) else error if format<-{enable Data format} then SetDataFormat(Curformat= format) else error if flow<-{enable flow control} then SetFlowControl(Flowcontrol= flow) else error if xchars<-{enable xchars} then SetXchars(Curxchars=xchars) else error if echars<-{enable echars} then SetEchars(Curechars=echars) else error IrCOMM_Control.request (Datarate=Currate, Dataformat=Curformat, Flowcontrol=Curflow, XON/XOFF=Curxchars, ENQ/ACK=Curechars) => CONNECTED

EVENT	STATE		
	IDLE	READY	CONNECTED
IrCOMM_Control.indication (break=status)	Error => IDLE	Error => READY	BS.request(status) => CONNECTED
DSR.indication(status)	if status = ready then => READY else => IDLE	if status = not ready then => IDLE else => READY	if status = not ready then DTR.request(not ready) IrCOMM_Disconnect.request (Reason='User Disconnect', Userdata=data) => IDLE else => CONNECTED
CD.indication(status)	=> IDLE	=> READY	if status = off then IrCOMM_Disconnect.request (Reason='User Disconnect', Userdata=data) DTR.request(not ready) =>READY else => CONNECTED
BS.indication(status)	=> IDLE	=> READY	IrCOMM_Control.request (Break = status) => CONNECTED
OE.indication	=> IDLE	=> READY	IrCOMM_Control.request (LineStatus = OverrunError) => CONNECTED
PE.indication	=> IDLE	=> READY	IrCOMM_Control.request (LineStatus = ParityError) => CONNECTED
FE.indication	=> IDLE	=> READY	IrCOMM_Control.request (LineStatus = FramingError) => CONNECTED
RI.indication(status)	=> IDLE	=> READY	=> CONNECTED

12.4.3.2 State Transition Table (Case.1 Flow Control Method: RTS/CTS flow control)

EVENT	STATE		
	IDLE	READY	CONNECTED
IrCOMM_Data.indication (data)	Error => IDLE	Error => READY	if DCEBusy = false then DT.request(data) => CONNECTED else put_into_DCEqueue(data) => CONNECTED
DT.indication(data)	Error ⇒ IDLE	Error ⇒ READY	if DTEBusy = false then IrCOMM_Data. request(data) => CONNECTED else put_into_DTEqueue(data) => CONNECTED
CTS.indication(status)	if status = ready then DCEBusy = false => IDLE else DCEBusy = true => IDLE	if status = ready then DCEBusy = false => READY else DCEBusy = true => READY	if status = busy then DCEBusy = true => CONNECTED else DCEBusy = false DT.request (get_from_DCEqueue) => CONNECTED
changeDTEstatus(status)	if status = ready then DTEBusy = false =>IDLE else DTEBusy = true =>IDLE	if status = ready then DTEBusy = false =>READY else DTEBusy = true => READY	if status = busy then DTEBusy = true RTS.request(busy) =>CONNECTED else DTEBusy = false IrCOMM_Data.request(get_from_DTEqueue) RTS.request(ready) =>CONNECTED

12.4.3.3 State Transition Table (Case.2 Flow Control Method: XON/XOFF flow control)

EVENT	STATE		
	IDLE	READY	CONNECTED
IrCOMM_Data.indication (data)	Error => IDLE	Error => READY	if data != XON ^ data != XOFF then if DCEBusy = false then DT.request(data) => CONNECTED else put_into_DCEqueue(data) => CONNECTED else if data = XOFF then DTEBusy = true DT.request(data) => CONNECTED else DTEBusy = false DT.request(data) IrCOMM_Data.request (get_from_DTEqueue) => CONNECTED
DT.indication(data)	Error => IDLE	Error => READY	if data != XON ^ data != XOFF then If DTEBusy = false then IrCOMM_Data. request(data) => CONNECTED else put_into_DTEqueue(data) => CONNECTED else if data = XOFF then DCEBusy = true => CONNECTED else DCEBusy = false DT.request (get_from_DCEqueue) => CONNECTED
CTS.indication(status)	=> IDLE	=> READY	=>CONNECTED

EVENT	STATE		
	IDLE	READY	CONNECTED
changeDTEstatus(status)	if status = ready then DTEBusy = false =>IDLE else DTEBusy = true =>IDLE	if status = ready then DTEBusy = false =>READY else DTEBusy = true => READY	if status = busy then DTEBusy = true RTS.request(busy) =>CONNECTED else DTEBusy = false IrCOMM_Data.request(get_from_DTEqueue) RTS.request(ready) =>CONNECTED

12.4.4

State Definitions

IDLE

The DCE is not ready (DSR = not ready).

READY

The DCE is active (DSR = ready) and a IrCOMM connection does not exist.

CONNECTED

The DCE is active (DSR = ready) and a IrCOMM connection exists.

12.4.5 State Variables

DCEBusy

If it is possible for IrTA to send data to DCE, DCEBusy is false. Otherwise this is true.

DTEBusy

If it is possible for IrTA to send data to IR-DTE, DTEBusy is false. Otherwise this is true.

12.4.6 Event Descriptions

LM_DiscoverDevices.indication

Receipt of a LM_DiscoverDevices.indication primitive from a LSAP connection endpoint.

IrCOMM_Connect.indication

Receipt of an IrCOMM_Connect.indication primitive from a IrCOMM SAP connection endpoint.

IrCOMM_Disconnect.indication

Receipt of an IrCOMM_Disconnect.indication primitive from a IrCOMM SAP connection endpoint.

IrCOMM_Control.indication

Receipt of an IrCOMM_Control.indication primitive from a IrCOMM SAP connection endpoint.

IrCOMM_Data.indication

Receipt of an IrCOMM_Data.indication primitive from a IrCOMM SAP connection endpoint.

DT.indication

Receipt of a DT.indication primitive from a DCE Service interface.

DSR.indication

Receipt of a DSR.indication primitive from a DCE Service interface.

CTS.indication

Receipt of a CTS.indication primitive from a DCE Service interface.

CD.indication

Receipt of a CD.indication primitive from a DCE Service interface.

BS.indication

Receipt of a BS.indication primitive from a DCE Service interface.

RI.indication

Receipt of a RI.indication primitive from a DCE Service interface.

OE.indication

Receipt of a OE.indication primitive from a DCE Service interface.

PE.indication

Receipt of a PE.indication primitive from a DCE Service interface.

FE.indication

Receipt of a FE.indication primitive from a DCE Service interface.

changeDTEstatus

Receipt of a changeDTEstatus primitive. This primitive is an internal event that indicates the change of the data transmitting conditions.

12.4.7 Action Descriptions

{Addr, Info, Method} = {Addr, Info, Method} U {addr, info, method}

Insert the information of the newly discovered station to that of the already-known stations.

IrCOMM_Connect.response

Send IrCOMM_Connect.response service primitive to the IrCOMM SAP connection endpoint.

IrCOMM_Disconnect.request

Send IrCOMM_Disconnect.request service primitive to the IrCOMM SAP connection endpoint.

IrCOMM_Data.request

Send IrCOMM_Data.request service primitive to the IrCOMM SAP connection endpoint.

DTR.request

Send DTR.request service primitive to the DCE Service interface.

RTS.request

Send RTS.request service primitive to the DCE Service interface.

BS.request

Send BS.request service primitive to the DCE Service interface.

put_into_DTEqueue

Put a data into the DTE sending queue.

put_into_DCEqueue

Put a data into the DCE sending queue.

get_from_DTEqueue

Get a data from the DTE sending queue (if this data exists).

get_from_DCEqueue

Get a data from the DCE sending queue (if this data exists).

Connected = ø

There are no IrCOMM connection at the IrCOMM SAP connection endpoint.

type <- {Service Type}

Service Type defined by IrCOMM_Connect.request is included in the service type possessed by IrTA.

General_Control_parameters = acceptable

IrCOMM is in the receiving state of IrCOMM_Control.request (Service Type) defined in section 4.5 of this document.

SetServicetype(Servicetype=type)

Set Servicetype of IrTA

SetDataRte(Currate=rate)

Set Current Data rate of a DCE Service interface

SetDataFormat(Curformat=format)

Set Current Data format of a DCE Service interface

SetFlowControl(Curflow=flow)

Set Current flow Control of a DCE Service interface

SetXchars(Curxchars=xchars)

Set Current XON/XOFF flow Control characters of a DCE Service interface

SetEchars(CurEchars=echars)

Set Current ENQ/ACK flow Control characters of a DCE Service interface

DCEBusy = false

Set false into the state variable DCEBusy.

DCEBusy = true

Set true into the state variable DCEBusy.

DT.request

Send DT.request service primitive to the DCE Service interface.

DTEBusy = false

Set false into the state variable DTEBusy.

DTEBusy = true

Set true into the state variable DTEBusy.

status = ready

The parameter status is ready.

status = not ready

The parameter status is not ready.

status = busy

The parameter status is busy.

status = off

The parameter status is off.

data != XON ^ data !=XOFF

data does not mean XON flow control character and XOFF flow control character.

data = XOFF

data mean XOFF flow control character.

rate<-{enable Data rate}

The rate is in the applicable area of Data rate in the IrTA.

format<-{enable Data format}

The format is one of the applicable format in the IrTA.

flow<-{enable flow control}

The flow is one of the applicable flow control in the IrTA.

xchars<-{enable xchars}

The xchars are the indictable XON/XOFF flow control characters.

echars<-{enable echars}

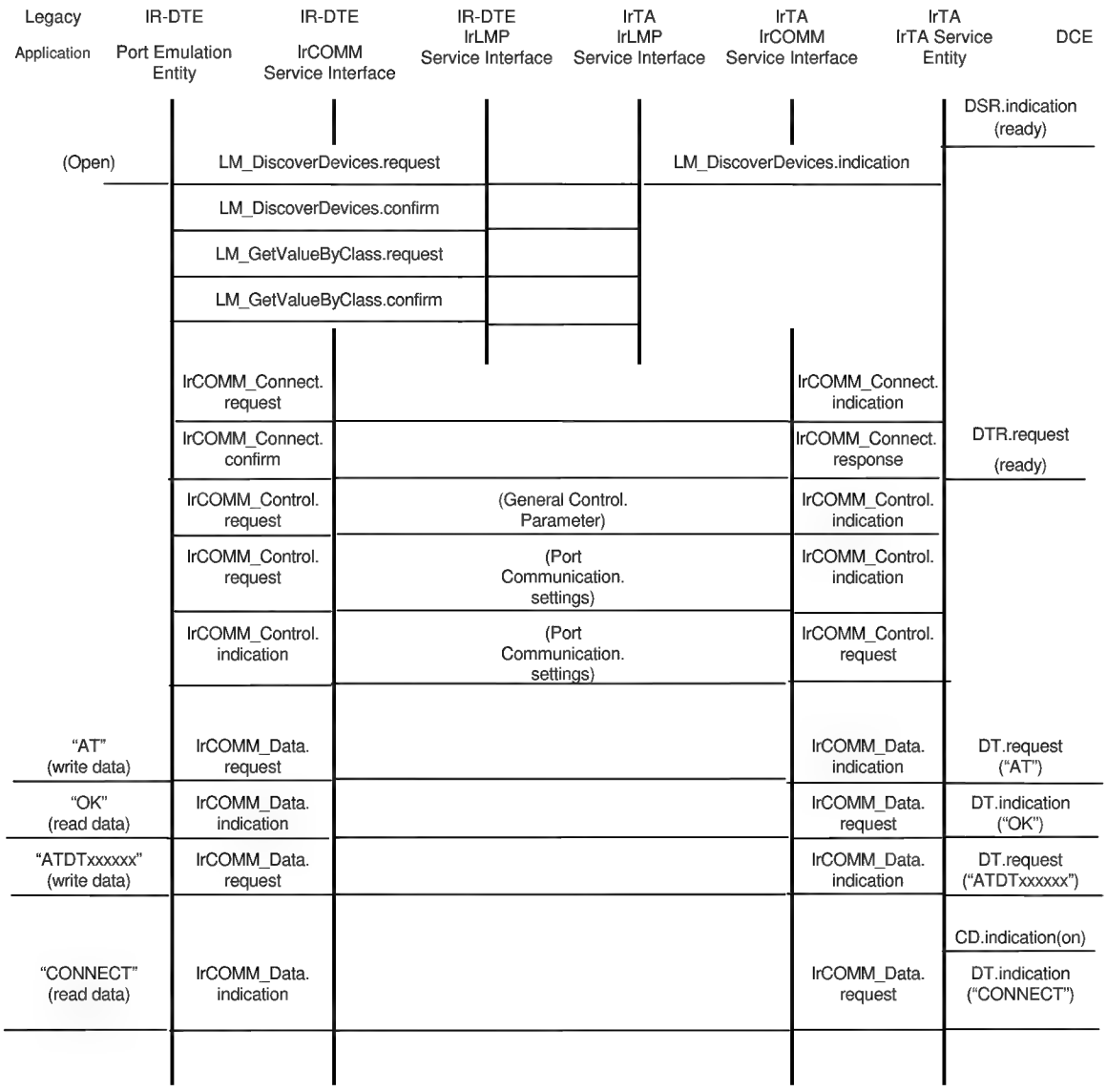
The echars are the indictable ENQ/ACK flow control characters.

Error

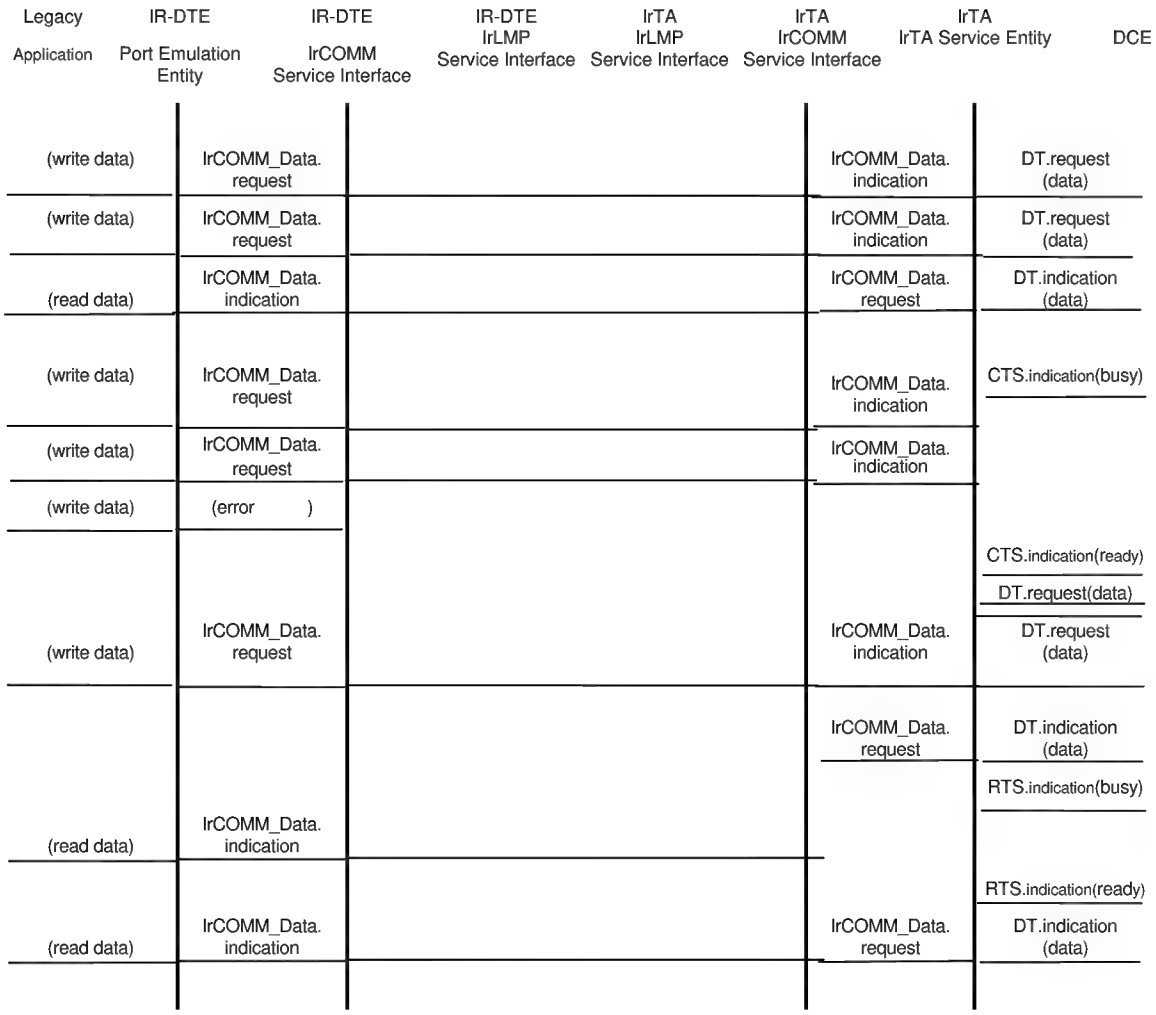
An unexpected or illegal event has occurred. These events are simply ignored in many cases.

12.5 IrTA Service Sequence Example

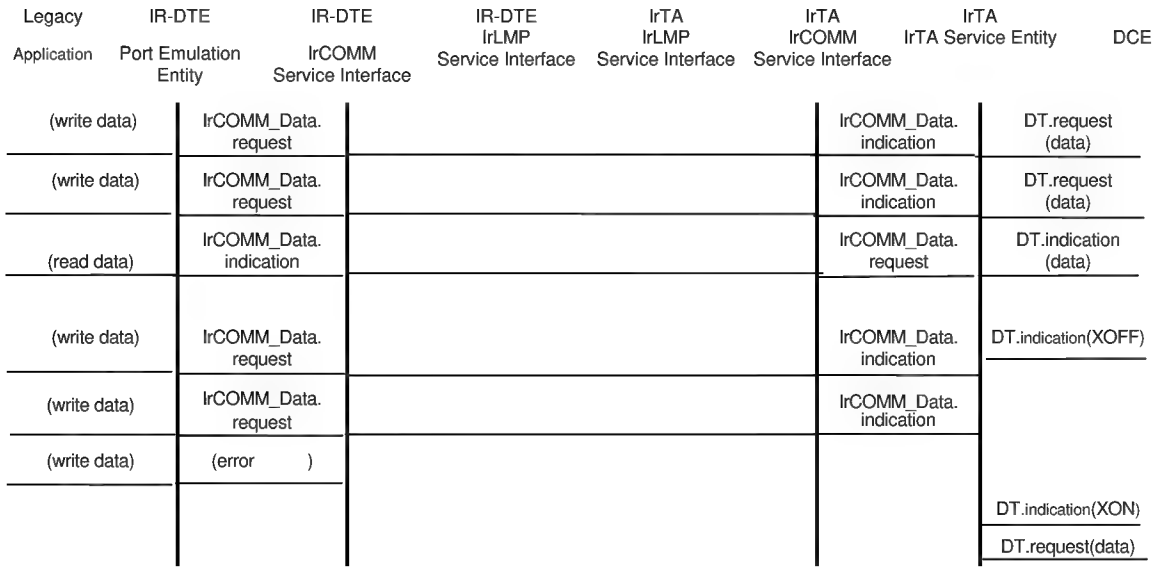
12.5.1 Normal (Call Connection Phase)

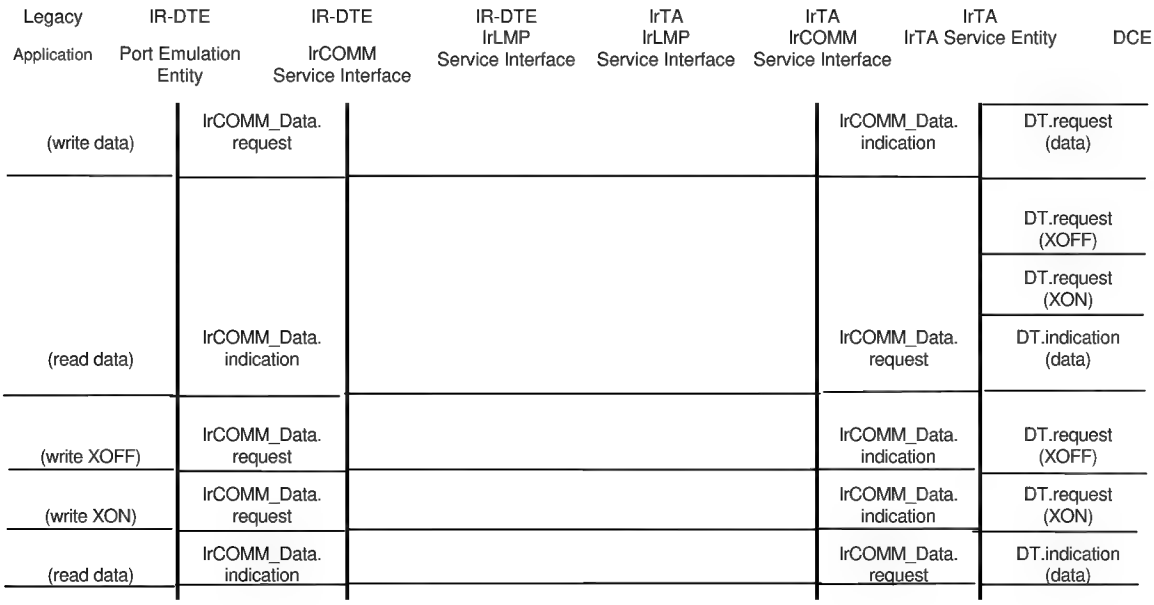


12.5.2 Normal(Data Transfer Phase; Flow control:RTS/CTS control)

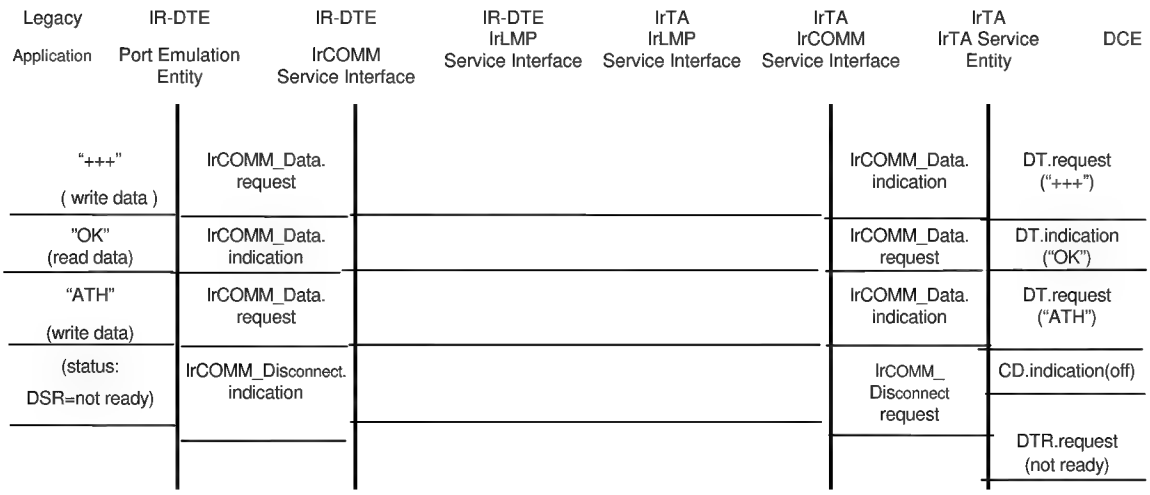


12.5.3 Normal(Data Transfer Phase; Flow control:XON/XOFF control)

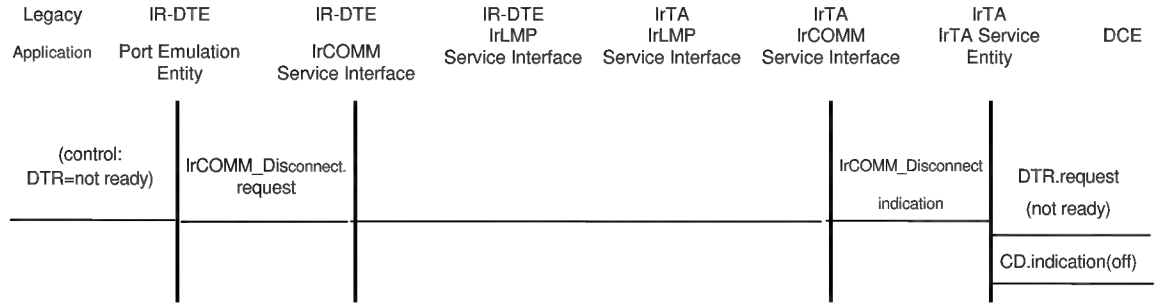




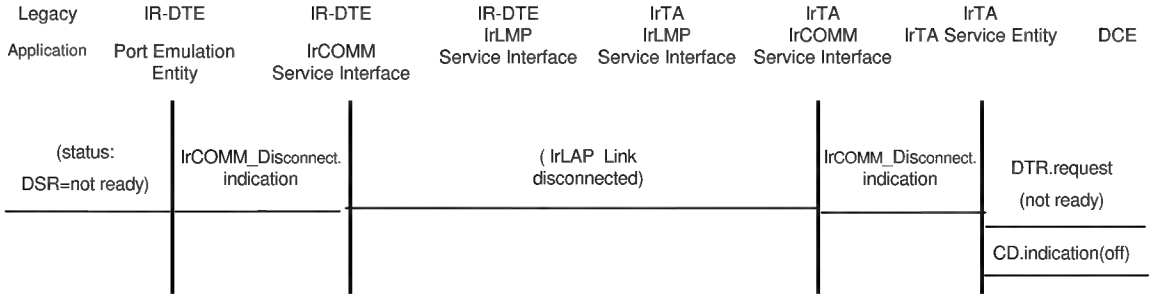
12.5.4 Normal (Call Disconnection Phase)



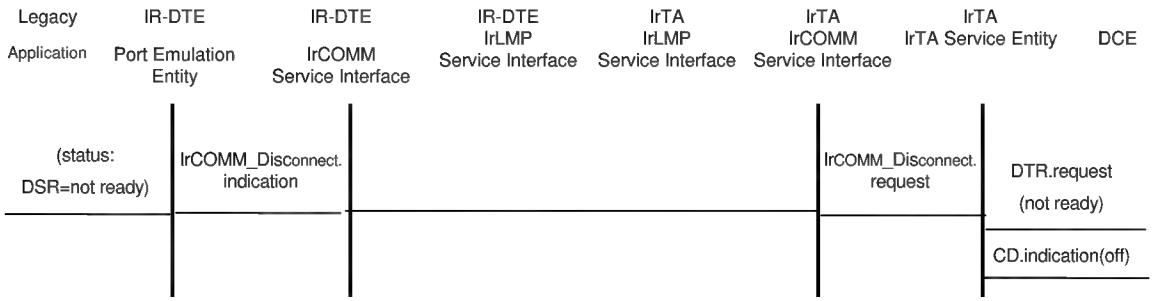
12.5.5 Abnormal(Enforced Disconnection from IR-DTE)



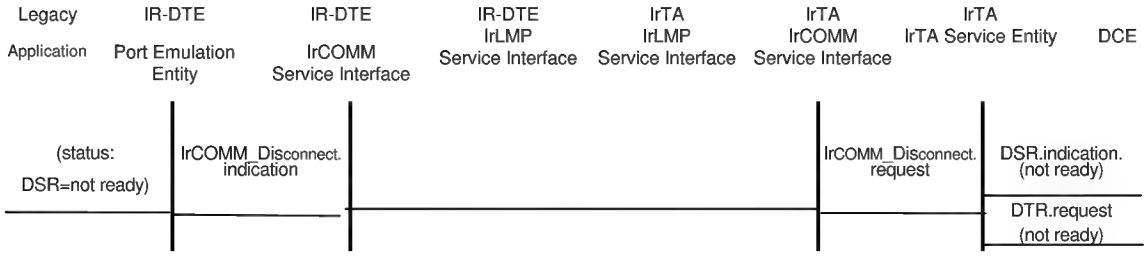
12.5.6 Abnormal(Abnormal Disconnection from IR Link)



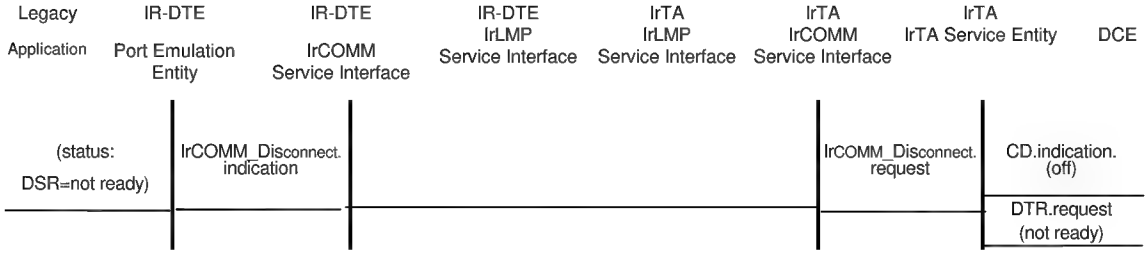
12.5.7 Abnormal(Abnormal Disconnection from IrTA)



12.5.8 Abnormal(Enforced Disconnection from DCE)



12.5.9 Abnormal (Enforced Disconnection Partner Terminal or Network)



12.6 Implementation alternative of IrTA and IR-DTE

12.6.1 IrTA procedure in the disconnection request from Network (PSTN/ISDN) via DCE

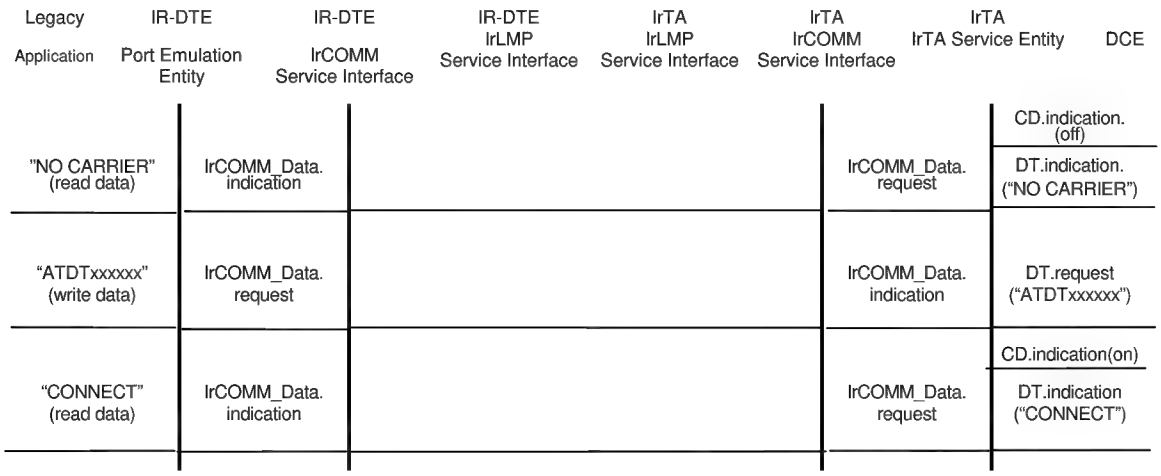
In this annex, when IrTA received CD.indication(off) from DCE(in the case of disconnection between DCE and network), is the matter of the implementation either IrTA will a) disconnecting the IrCOMM link, or b) holding the IrCOMM link.

The state transition in the clause 12.4 and service sequence examples in the clause 12.5 describe in the case a).

In the case a), the state of CONNECTED means that the connection of IrCOMM and between DCE and network are holding. When it transit to the state of READY or IDLE, IrCOMM link and the connection between DCE and network are disconnected.

In the case b), regardless of DCE connection with network, the state of CONNECTED means that IrCOMM link is established. In this case, Port Emulation Entity has no way how the status of Legacy Application are. Therefore, in the DCE, the information of the character data from DCE (e.g. *NO CARRIER*, etc.) indicating line disconnection, is necessary.

The following shows an example for service sequence in the case b).



12.6.2 IrTA procedure when the IrCOMM disconnected during the connection phase between DCE and network (PSTN/ISDN)

During the connection between DCE and network (CONNECTED), when DCE received IrCOMM_Disconnect from IrCOMM by the request from Port Emulation Entity of IR-DTE or time-out, IrTA will choose one of the following cases.

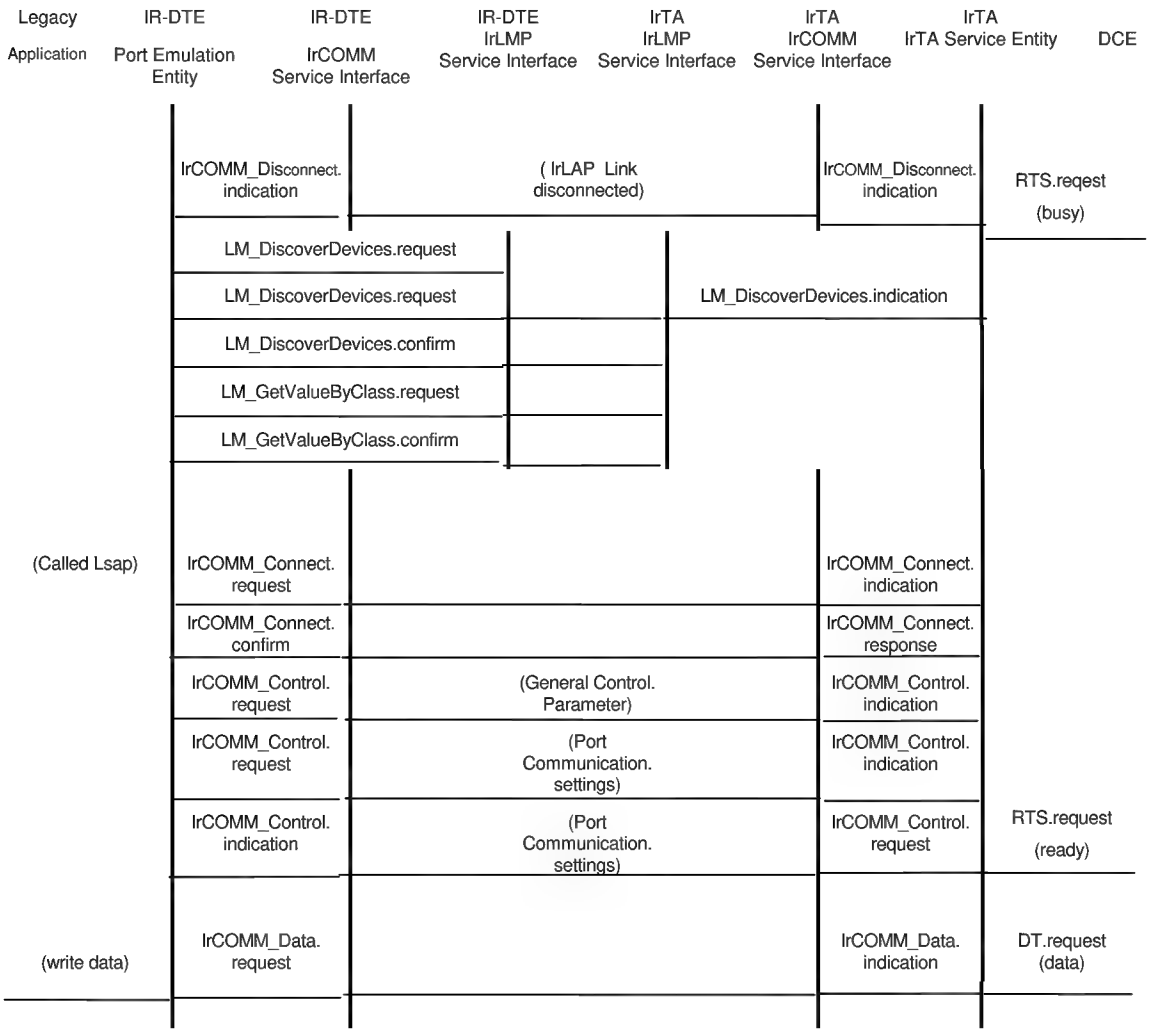
a) IrTA send DTR.request(off) to DCE, and disconnect the network, then makes DCE to transit to the "READY status" (state transition and service sequence are the same as before).

b) In the case of the reason for IrCOMM_Disconnect mean "User Disconnect", it is the same as the case a). But in the case of "Provider Disconnect" (e.g. line disconnection by transmission problem, etc.), IrTA execute the flow control to DCE (e.g., in the case of RTS/CTS control, IrTA send RTS.request (busy), and in the case of XON/XOFF control, it send DT.request (XOFF)), then wait a fixed time until establishment of IrCOMM link. When timer is expired(Time-out), IrTA send DTR.request(not ready) to

DCE. If the link is re-established during timer is not expired, IrTA cancel the flow control, and it is continue the communication(in the case of RTS/CTS IrTA send RTS.request(ready), and in the case of XON/XOFF control, it send DT.request(XON)).

An example of service sequence of case b) is shown as follows.

Here, in the case of b) in the state transition, mentioned above, the addition of the state of SUSPEND(waiting IrCOMM link re-establishment) is necessary.



12.6.3 Start of the establishment of the IrCOMM link

After the finding of IrTA using LM_DiscoverDevices by the Port Emulation Entity of IR-DTE, the IrCOMM link is established as follows.

a) when Legacy Application execute the port open (initialize the serial port, and set the possible communication parameters), establish the IrCOMM link.

After Legacy Application execute the port open, when the IrCOMM link is disconnected, it close the port. The re-establishment of the IrCOMM link is done by the port open again.

(the same as the sequence of 12.5)

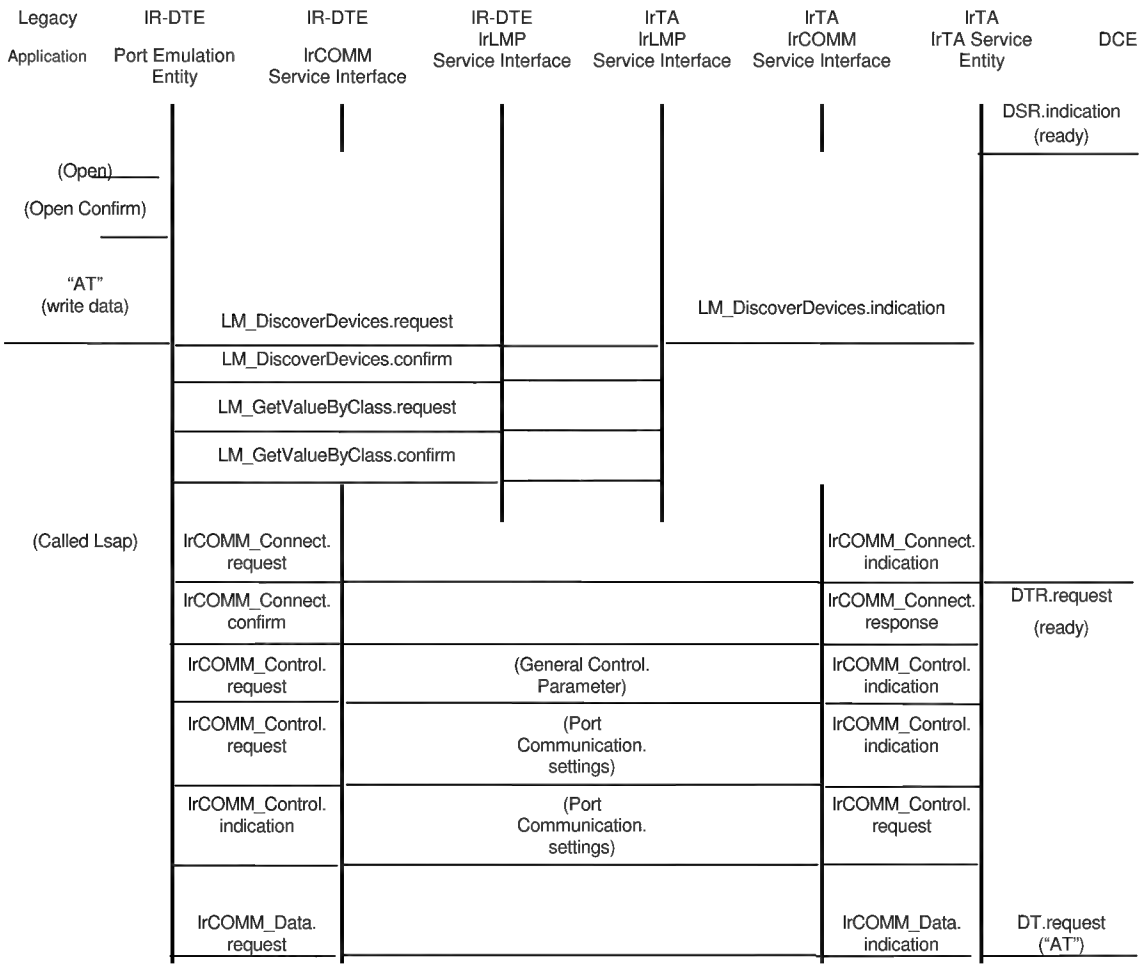
b) when Legacy Application execute the port open, IrCOMM link is established.

When the IrCOMM link was disconnected without port closing, the Legacy Application execute the LM_DiscoverDevices again, and establish the IrCOMM link again.

(In the case of link disconnection, it is the same as the sequence of 12.6.2.)

c) At the time of the port open, the port emulation entity should not establish the IrCOMM link, when it received the data to be transmitted from the Legacy Application, if the IrCOMM link is not established, it establish the IrCOMM link.

An example of service sequence of case c) is shown as follows.



In the case of a), it is necessary for Legacy application to open and close a port every time **it** connect with IrTA.

On the other hand, in the case of b) and c), port is always open and is able to execute the re-establishment procedure, mentioned above.

However, if an IrTA to be connected is not discovered, it should periodically send LM_DiscoverDevices until an IrTA is found or it times out (case b and c).

12.6.4 Treatment of Break signal

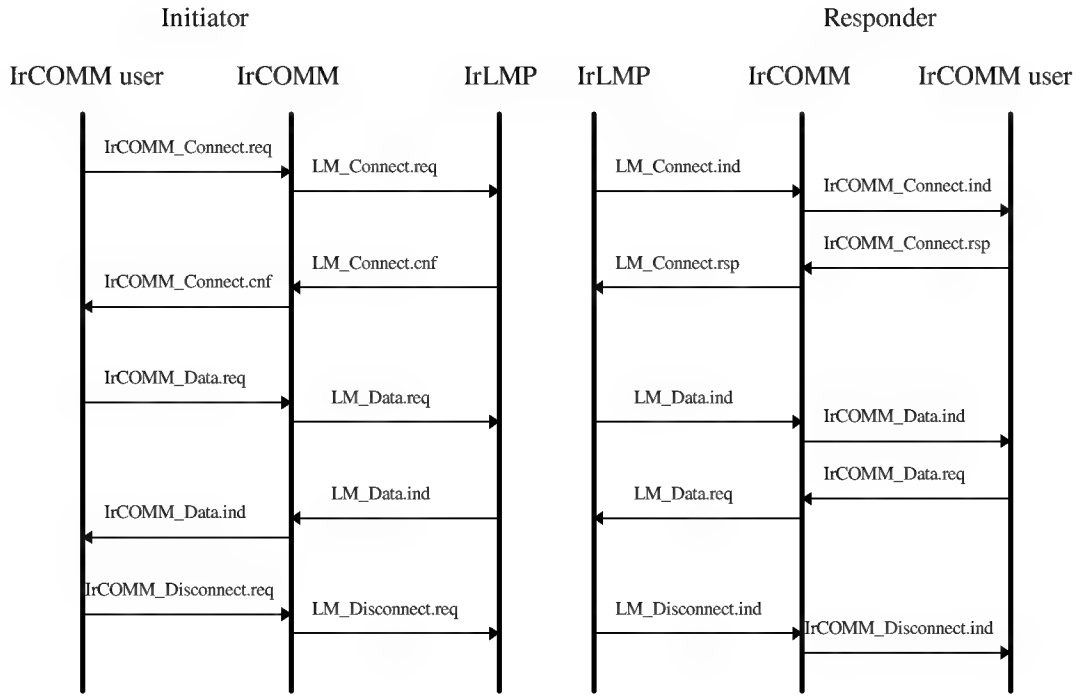
Break signal will be transmitted on the data line, therefore during the signal transmission, all signals except data signal may not be reliable.

At the time of the Break signal request, if the Port Emulation Entity or IrTA received some data already, there are some choice as follows. The selection is the matter of the implementation.

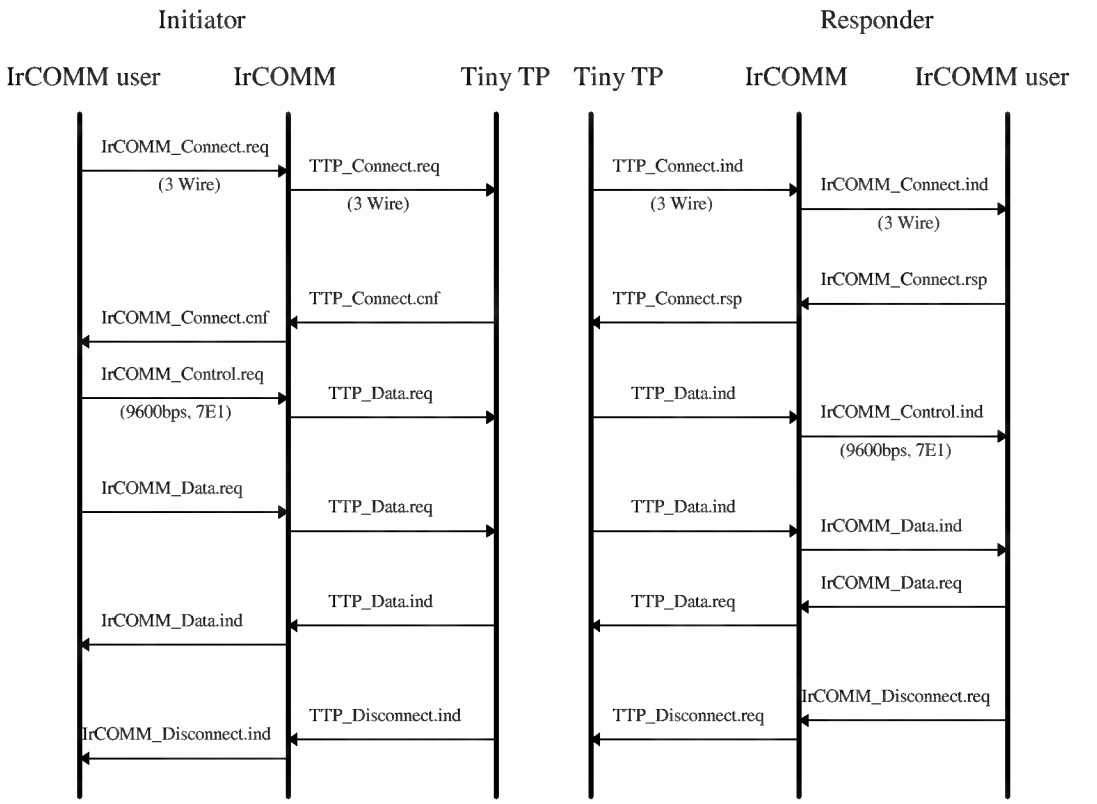
(a) discard the Break signal, (b) process the Break signal, (c) process the Break signal, but the data should be kept. (Reference [ITU-TV.42] clause 7.4, 7.5)

13. APPENDIX A. Typical protocol sequence examples

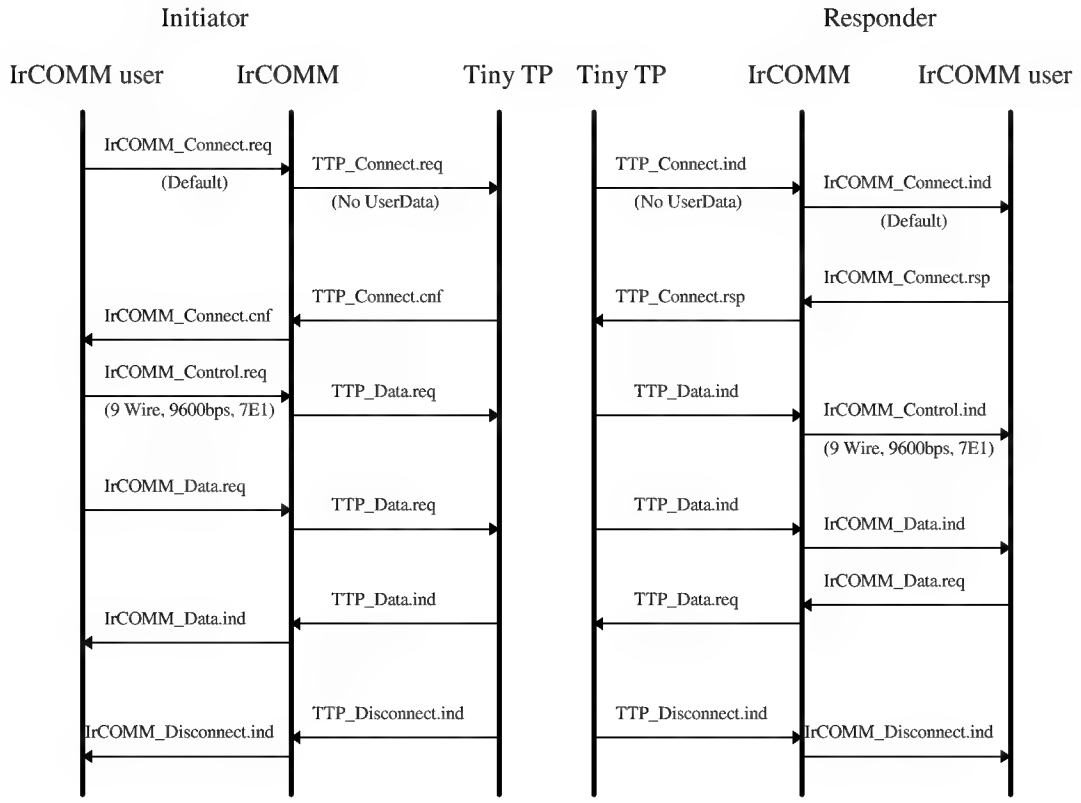
13.1 3-Wire raw



13.2 3-Wire, Service type is sent by TTP_Connect



13.3 9-Wire, Service type is sent by TTP_Data



14. APPENDIX B Interfacing IrCOMM to Data or Fax Modem

The purpose of this appendix is to define the functionality of the IrCOMM when being used with data or fax modems or other equipment capable of acting as such devices. For most parts only the external behavior is defined to ensure that a great variety of data and fax modem implementations and other kinds of devices can exist. The internal organization, or definite FSMs, are left open.

The main body of the IrCOMM specification defines the underlying protocol frame format on which the functionality of IrCOMM is based. In this appendix the functionality implemented on top of that framework is further defined to a degree, that allows one to rely on the external behavior of an IrCOMM instance.

This material is intended to be used with the 9-wire service type. Some sections are applicable also to 3-wire, and even to 3-wire RAW service types.

14.1 Naming, References to External Entities

- Client refers to the local user of the local IrCOMM instance. Data flow from client means the characters that are written to the IrCOMM. Data flow to client means the characters that the IrCOMM writes to its client.
- Host refers to the remote IrCOMM instance. (i.e. The IrCOMM instance on the other end of the IR connection). Data flow from host means the characters received from the IrCOMM instance at the other end of the connection. Data flow to host means the characters that are sent to the IrCOMM instance at the other end of the connection.

The naming convention is shown in Figure 1.

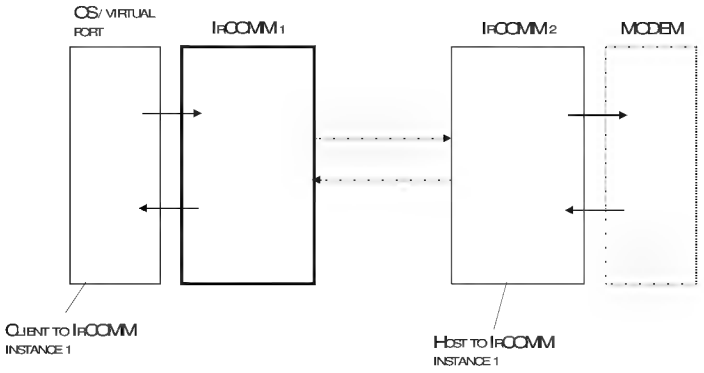


Figure 1. The naming convention

14.2 External Interfaces

The interface between the IrCOMM and its client is very much dependent on the system in which the IrCOMM instance is implemented. This appendix does not define the interface, though a list of the information exchanged in this interface is given.

The information exchanged at the client interface of the IrCOMM instance is the following:

i. Communication Settings (to and from the client). Communication Settings consist of data rate, data format, flow control method, and XON/XOFF flow control characters.

ii. Line Status information (to and from client). This consists of the Overrun, Parity, and Framing error indications).

iii. Data flows (to and from the client).

iv. ITU-T V.24 signals (to and from client). The direction of the signals depends on the client type, DTE or DCE.

v. Break signal (to and from client). The break signal may be transferred directly through the IrCOMMs though some implementations may have additional functionality.

Figure 2 shows the information exchanged between IrCOMM instances and their clients.

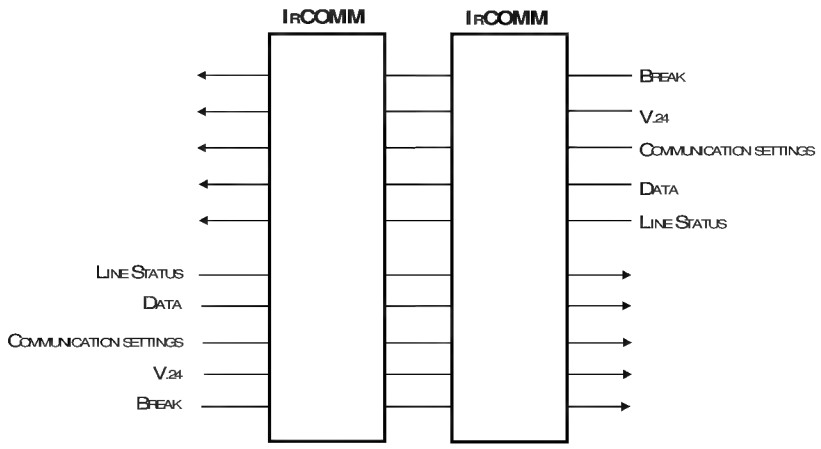


Figure 2. An overview on the client interface.

14.3 Flow Control

IrCOMM is required to locally emulate the flow control method in effect. It must also be able to detect the situation in which the client of IrCOMM (for example, local virtual communication port) activates flow control, and to act accordingly. If both IrCOMM connection endpoints are studied, there is four individual cases which must each be dealt with:

i. The DTE is unable to accept more data from the local IrCOMM instance, and it is required to control the flow off from the IrCOMM instance. The IrCOMM must be able to detect the signaling the DTE uses for flow control.

ii. The IrCOMM instance is unable to accept more data from the DTE, and it is required to control the flow off from the DTE. The IrCOMM must be able use a flow control method that the DTE can identify.

iii. The DCE is unable to accept more data from the local IrCOMM instance, and it is required to control the flow off from the IrCOMM instance. The IrCOMM must be able to detect the signaling the DCE uses for flow control.

iv. The IrCOMM instance is unable to accept more data from the DCE, and it is required to control the flow off from the DCE. The IrCOMM must be able use a flow control method that the DCE can identify.

In existing RS-232 port based systems the method that is used for flow control may vary from system to system. Though for data and fax modems just the two mutually nonexclusive flow control methods, XON/XOFF and RTS/CTS, are used, this appendix describes also the functionality of the DSR/DTR flow control.

The XON/XOFF flow control is generally called software flow control. RTS/CTS and DSR/DTR based methods are called hardware flow control methods. The flow control methods are not mutually exclusive, that is, the software flow control may be active at the same time with the hardware flow control method.

14.3.1 XON/XOFF flow control

The IrCOMM instance is required to detect XON/XOFF characters coming from its client when XON/XOFF flow control on input is enabled. In this case, the XON/XOFF characters are acted upon, but not forwarded to the host IrCOMM instance. If the XON/XOFF flow control is disabled on input, these characters are processed as normal data. Any XON/XOFF characters received from the host are processed as normal data. If the XON/XOFF flow control is enabled on output, then the IrCOMM instance may use XON and XOFF characters to exercise flow control to the client. If XON/XOFF flow control is supported, it is recommended that it is enabled on both input and output of the IrCOMM entities.

14.3.2 RTS/CTS flow control

The IrCOMM instance is required to detect RTS/CTS signal changes (depending on the type of its client, respectively DTE or DCE) signaled by its client when RTS/CTS flow control on input is enabled. The signal changes are acted upon, and forwarded to the host IrCOMM instance. If the RTS/CTS flow control is enabled on output, then the IrCOMM instance may use changing of the RTS/CTS state to exercise flow control to the client. If RTS/CTS flow control is supported, it is recommended that it is enabled on both input and output of the IrCOMM entities.

When hardware flow control, using signals RTS and CTS, is in effect, then the functionality of the modem interface is the following:

- DSR, DCD, and RI signals are transferred without interception from the DCE to the DTE.
- DTR signal is transferred without interception from the DTE to the DCE.
- RTS and CTS signals are used to implement the local flow control.

This can be seen in the Figure 3.

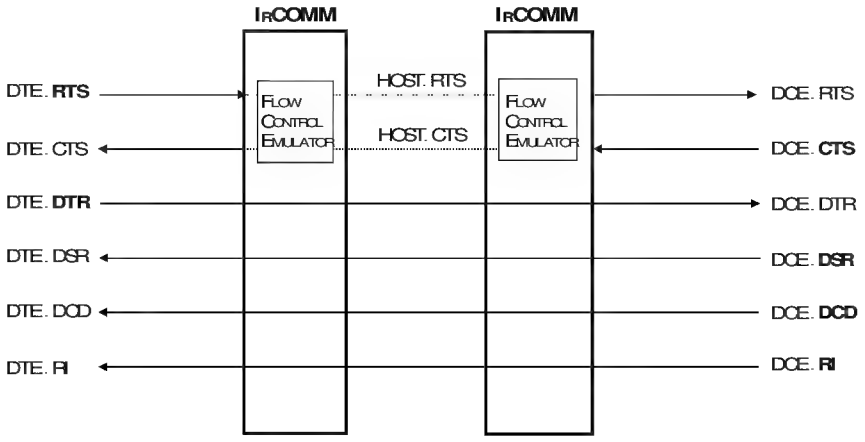


Figure 3. The RTS/CTS flow control V.24 signals

The functions of the flow control circuits can be implemented in the following way:

The DTE side / Client indicates flow control:

DTE sets	Local IrCOMM interpretation	Host IrCOMM sees
DTE.RTS is 'ON'	IrCOMM may send data to client	HOST.RTS := 'ON'
DTE.RTS is 'OFF'	IrCOMM may not send data to client	HOST.RTS := 'OFF'

The DTE side / IrCOMM indicates flow control:

Local IrCOMM sets	DTE sees
IrCOMM flow control active	DTE.CTS := HOST.CTS AND 'OFF'
IrCOMM flow control off	DTE.CTS := HOST.CTS AND 'ON'

The DCE side / Client indicates flow control:

DCE sets	Local IrCOMM interpretation	Host IrCOMM sees
DCE.CTS is 'ON'	IrCOMM may send data to client	HOST.CTS := 'ON'
DCE.CTS is 'OFF'	IrCOMM may not send data to client	HOST.CTS := 'OFF'

The DCE side / IrCOMM indicates flow control:

IrCOMM sets	DCE sees
IrCOMM flow control active	DCE.RTS := HOST.RTS AND 'OFF'
IrCOMM flow control off	DCE.RTS := HOST.RTS AND 'ON'

This is not the only possible way of generating the flow control signals. In some implementations there is the possibility to take control of the flow control signals totally. This means that even though the host system indicates that flow control is active (i.e. RTS/CTS signals is not asserted), the local IrCOMM instance asserts the signal to enable its client to transmit data to it. This is mainly done to ensure that the IrCOMM buffers do not become empty due to local flow control indications that are seen by the client of the host IrCOMM instance.

14.3.3 DSR/DTR flow control

The IrCOMM instance is required to detect DSR/DTR signal changes (depending on the type of its client, respectively DTE or DCE) signaled by its client when DSR/DTR flow control on input is enabled. The signal changes are acted upon, and forwarded to the host IrCOMM instance. If the DSR/DTR flow control is enabled on output, then the IrCOMM instance may use changing of the DSR/DTR signal level to exercise flow control to the client. If DSR/DTR flow control is supported, it is recommended that it is enabled on both input and output of the IrCOMM entities.

When hardware flow control, using signals DSR and DTR, is in effect, then the functionality of the modem interface is the following:

- CTS, DCD, and RI signals are transferred without interception from the DCE to the DTE.
- RTS signal is transferred without interception from the DTE to the DCE.
- DSR and DTR signals are used to implement the local flow control.

This can be seen in the Figure 4.

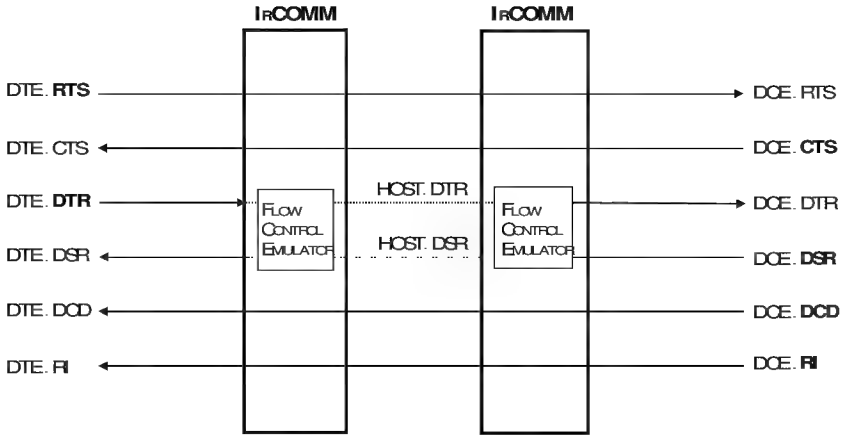


Figure 4. The DSR/DTR flow control V.24 signals

The functions of the flow control circuits can be implemented in the following way:

The DTE side / Client indicates flow control:

DTE sets	Local IrCOMM interpretation	Host IrCOMM sees
DTE.DTR is 'ON'	IrCOMM may send data to client	HOST.DTR := 'ON'
DTE.DTR is 'OFF'	IrCOMM may not send data to client	HOST.DTR := 'OFF'

The DTE side / IrCOMM indicates flow control:

Local IrCOMM sets	DTE sees
IrCOMM flow control active	DTE.DSR := HOST.DSR AND 'OFF'
IrCOMM flow control off	DTE.DSR := HOST.DSR AND 'ON'

The DCE side / Client indicates flow control:

DCE sets	Local IrCOMM interpretation	HOST IrCOMM sees
DCE.DSR is 'ON'	IrCOMM may send data to client	HOST.DSR := 'ON'
DCE.DSR is 'OFF'	IrCOMM may not send data to client	HOST.DSR := 'OFF'

The DCE side / IrCOMM indicates flow control:

IrCOMM sets	DCE sees
IrCOMM flow control active	DCE.DTR := HOST.DTR AND 'OFF'
IrCOMM flow control off	DCE.DTR := HOST.DTR AND 'ON'

Again, this is not the only possible way of generating the flow control signals. In some implementations there is the possibility to take control of the flow control signals totally. This means that even though the host system indicates that flow control is active (i.e. DSR/DTR signals is not asserted), the local IrCOMM instance asserts the signal to enable its client to transmit data to it. This is mainly done to ensure that the IrCOMM buffers do not become empty due to local flow control indications that are seen by the client of the host IrCOMM instance.

14.4 Procedures for Changing Communication Settings

The main body of the IrCOMM specification defines that when ever the Communication Settings change, they have to be reported to the host IrCOMM instance. Nevertheless, the host system is not required explicitly to act upon these changes.

In this sections a recommended way to deal with the change of Communication Settings is given. The method should enable existing communication applications to work correctly, as well as, it should make it possible to interface the IrCOMM to data services in GSM/PCS/ISDN networks.

Requirements for DTE:

When Communication Settings change indication is received from the DCE, the DTE should do one of the following:

i. Do not change it's settings and do not send a communication settings change indication to the DCE.

ii. Change it's settings to match those of the DCE and send a communication settings change indication to the DCE with the new parameters that are in effect.

Requirements for DCE:

When communication settings change indication is received from the DTE, the DCE should change its settings to correspond those received from the DTE. The DCE should NOT generate a communication settings change indication due to a conflict in the settings. It should either apply the settings received from the DTE or to disconnect the service.

The only exception to these rules is the connection establishment. When communication settings are received from the DTE, the DCE responds either with the same settings or with new settings. (The new settings may apply to a received call that has been received by the DCE).

The connection establishment in the normal situation is shown in Figure 5. The DCE replies with the same InitialControlParameters to the DTE that it received from it. The Communication Settings of the DCE are totally controlled by the DTE.

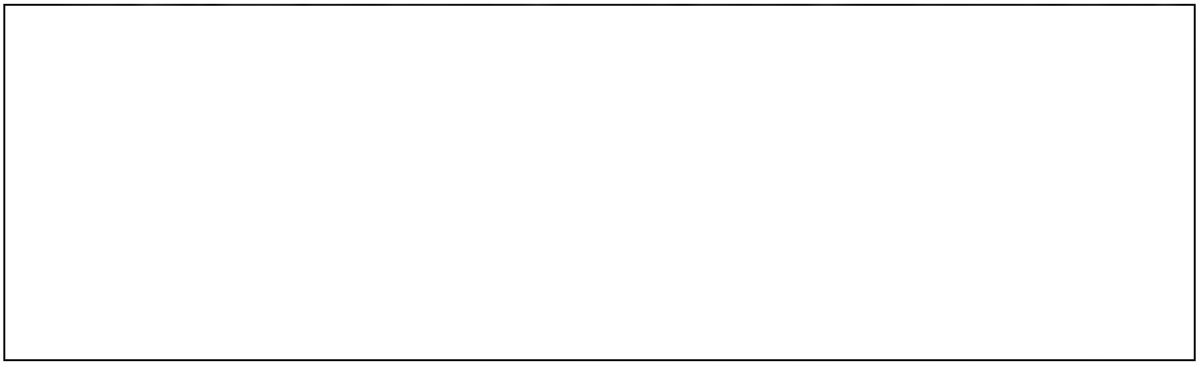


Figure 5. The normal connection establishment

The second scenario, in Figure 6, is related to secondary initiated connection establishment that will be discussed in detail later in this appendix. The secondary with DCE functionality is able to report the Communication Settings it has received from the network.

Figure 6. The Communication Settings for secondary initiated connection establishment

14.5 Internal Organization of IrCOMM

The recommended internal organization of the IrCOMM for data and fax modems consists of at least two FSMs, which are Client Control FSM and Host Control FSM. There may be other components present such as the built-in transport (TinyTP). The internal organization is shown in Figure 7.

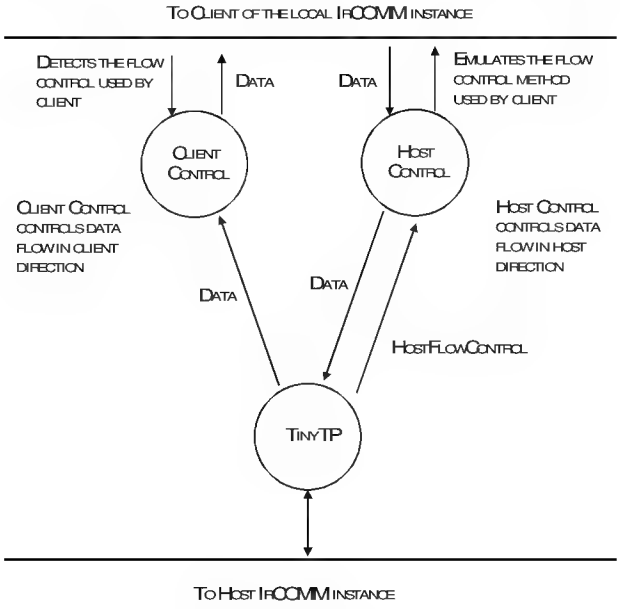


Figure 7. The Internal Organization

The internal organization except for Client Control FSM and Host Control FSM, is left open. The other entities are such as protocol encoder and decoder, the TinyTP, etc.

14.6 IrCOMM Client Control

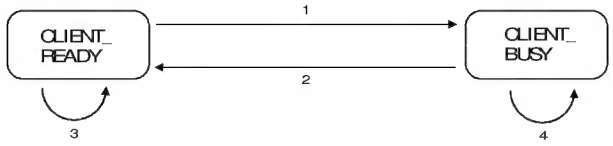
14.6.1 Purpose

It is recommended that the IrCOMM Client Control is present in every IrCOMM instance that is located in a system with data or fax modem functionality. The responsibility of the Client Control is to detect flow control signal changes and method changes, and to detect when the client of the IrCOMM (local VCOMM for example) is unable receive data from the IrCOMM instance.

14.6.2 Overview

In the CLIENT_READY state the Client Control FSM is able to send any data received from the host IrCOMM instance to its client. Transition to CLIENT_BUSY state occurs when the client has indicated that flow control is activated. In CLIENT_BUSY state the IrCOMM instance waits for the client to indicate that the flow control is off, buffering any data received from the host IrCOMM instance. When the client turns the flow control off, the Client Control will hand any buffered data to the client, and change state back to CLIENT_READY.

14.6.3 IrCOMM Client Control State Transition Diagram



14.6.4 IrCOMM Client Control State Transition Table

State	Event	Action	Next State	
CLIENT_READY	Data received from host	Forward data to client	CLIENT_READY	3
	XON/XOFF on input and XOFF received from client		CLIENT_BUSY	1
	RTS/CTS on input AND RTS/CTS ON-to-OFF transition from client		CLIENT_BUSY	1
	DSR/DTR on input AND DSR/DTR ON-to-OFF transition from client		CLIENT_BUSY	1
CLIENT_BUSY	Data received from host	Append to internal client data buffer	CLIENT_BUSY	4
	XON/XOFF on input AND XON received	Forward buffered client data to client	CLIENT_READY	2
	RTS/CTS on input AND RTS/CTS OFF-to-ON transition	Forward buffered client data to client	CLIENT_READY	2
	DSR/DTR on input AND DSR/DTR OFF-to-ON transition	Forward buffered client data to client	CLIENT_READY	2

14.7 IrCOMM Host Control

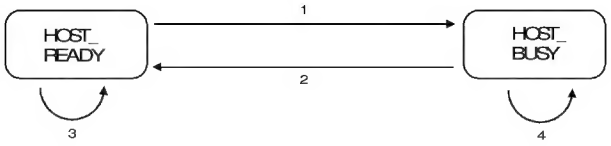
14.7.1 Purpose

It is recommended that the IrCOMM Host Control is present in every IrCOMM instance that is located in a system with data or fax modem functionality. The responsibility of the Host Control is to detect flow control state changes between the two IrCOMM instances, and to emulate the flow control in the method indicated by flow control method settings. This will enable the IrCOMM instance to flow off its client when it is unable to transmit data to the host IrCOMM instance due to lack of buffering space.

14.7.2 Overview

In the HOST_READY state the Host Control FSM is able to send any data received from its client to the host IrCOMM instance. Transition to HOST_BUSY state occurs when the flow control is activated between the IrCOMM instances. The Host Control FSM will emulate the flow control settings the client is able to detect to flow off the client. In the HOST_BUSY state the IrCOMM waits for the flow control between the IrCOMM instances to be turned off. When this happens, the Host Control FSM will again emulate the flow control method of the client to disable the flow control, and a state transition to HOST_READY occurs. In the HOST_BUSY state the IrCOMM instance may either discard any data received from its client, or buffer the data to be forwarded to the host when transition to HOST_READY state takes place.

14.7.3 IrCOMM Host Control State Transition Diagram



14.7.4 IrCOMM Host Control State Transition Table

State	Event	Action	Next State	
HOST_READY	Data received from client	Forward data to host	HOST_READY	3
	HostFlowControl AND XON/XOFF on output	Send XOFF to client ActiveMethod := 'XON/XOFF'	HOST_BUSY	1
	HostFlowControl AND RTS/CTS on output	RTS/CTS ON-to-OFF transition to client ActiveMethod := 'RTS/CTS'	HOST_BUSY	1
	HostFlowControl AND DSR/DTR on output	DSR/DTR ON-to-OFF transition to client ActiveMethod := 'DSR/DTR'	HOST_BUSY	1
HOST_BUSY	Data received from client	Discard data	HOST_BUSY	4
		Append to internal host data buffer		
	NOT HostFlowControl AND ActiveMethod := 'XON/XOFF'	Send XON to client	HOST_READY	2
		Send XON to client Forward buffered host data to host		
	NOT HostFlowControl AND ActiveMethod := 'RTS/CTS'	Client RTS/CTS := Host RTS/CTS (May cause OFF-to-ON transition to client)	HOST_READY	2
		Client RTS/CTS := Host RTS/CTS (May cause OFF-to-ON transition to client) Forward buffered host data to host		
	NOT HostFlowControl AND ActiveMethod := 'DSR/DTR'	Client DSR/DTR := Host RTS/CTS (May cause OFF-to-ON transition to client)	HOST_READY	2
		Client DSR/DTR := Host RTS/CTS (May cause OFF-to-ON transition to client) Forward buffered host data to host		

HostFlowControl Boolean value for the status of the flow control between the local and host IrCOMM instances. May be derived from the TinyTP.

ActiveMethod The flow control method which was used to emulate client flow control.

14.8 DCE Initiated Connection Establishment - Incoming Call

When a call is received from the network by a DCE with no active connection it is recommended to function in the way described here. Due to the fact that data and fax modems will be for most part secondary only devices, the method is based on functionality that can be made easily available in that kind of devices. It should be noted though, that this method is not limited to secondary only devices or to a specific service.

14.9 Requirements

The method requires sniffing functionality, and a specific usage of the hints fields in the sniffing frame. It is recommended that during the sniffing, only the hints bits of services requesting the connection are set. This will enable the primary to decide if it wants the service connection to be established depending on the services the secondary announces.

14.10 Functionality

The DCE reacts to the incoming call in the following way:

i. Sets the hint bit indicating IrCOMM service in the hint bits, clearing others. (PnP support bit may be set though)

ii. Initiates sniffing procedure.

iii. If sniffing is not successful, DCE drops the call (if the call was answered).

Device with IrCOMM instance and DTE functionality

i. Detects the sniffing request.

ii. Establishes a connection with the sniffing device if the sniffing device supports the service category the primary requires, in this case it is the IrCOMM service.

iii. Makes an IAS query for the IrCOMM parameters and for the LSAP selector.

iv. Establishes a service-to-service connection by binding the local and host LSAPs.